BASIC D'UN SEUL COUP D'ŒIL

Les instructions ci-dessous figurent dans le Manuel de Référence BASIC TO7

ABS
AND
ASC
ATTRB
AUTO

BEEP
BOX
BOXF

CDBL
CHR$
CINT
CLEAR
CLOSE
CLS
COLOR
CONSOLE
COS
CSNG
CSRLIN

DATA
DEFDBL
DEFGR$
DEFINT
DEFSNG
DEFSTR
DEFUSR
DELETE
DIM

ELSE
END
EOF
EQV
ERL
ERR
ERROR
EXEC
EXP

FIX
FOR ... NEXT
FRE

GOSUB ... RETURN

GR$

GOTO
HEX$

IF ... THEN ... ELSE
IMP
INKEY$
INPEN
INPUT
INPUT #
INPUT$
INPUTPEN
INPUTWAIT
INSTR
INT

LEFT$
LEN
LET
LINE
LINEINPUT
LINEINPUT #
LIST
LOAD
LOADM
LOCATE
LOG

MERGE
MID$
MOD
MOTOROFF
MOTORON

NEXT
NEW

OCT$
ON ERROR GOTO
ON ... GOSUB ...
ON ... GOTO ...
ONPEN GOSUB
ONPEN GOTO
OPEN
OR

PEEK
PEN
PLAY
POINT

POKE
POS
PRINT
PRINT #
PRINT # USING
PRINT USING
PSET
PTRIG

READ
REM
RESTORE
RESUME
RETURN
RIGHT$
RND
RUN

SAVE
SAVEM
SCREEN
SCREEN PRINT
SGN
SIN
SKIPF
SPC
SQR
STEP
STICK
STOP
STR$
STRIG

TAB
TAN
THEN
TROFF
TRON

UNMASK
USING
USR

VAL
VARPTR

WAIT

XOR

INITIATION AU BASIC TO7

Christine et François-Marie Blondel

Illustrations : Dominique Izoard



Dans la même collection

Le BASIC D.O.S. du TO7/TO7-70 et du MO5 — Christine et François-Marie Blondel

Un ordinateur à la maison — Jean Delcourt

Un ordinateur en fête — Serge Pouts-Lajus.

Un ordinateur et des jeux — Jean-Pascal Duclos.

Guide pratique de l'ordinateur personnel d'IBM — Dalloz/Emery/Portefaix/Boisgontier/Salzman

LOGO, des ailes pour l'esprit — Horacio C. Reggini

Premiers pas avec le ZX-SPECTRUM — Ian Stewart/Robin Jones

Le langage machine de ZX-SPECTRUM — Ian Stewart/Robin Jones

Plus loin avec le ZX-SPECTRUM — Ian Stewart/Robin Jones

Jeux vidéo, jeux de demain — Georges-Marie Becherraz/Alain Graber

Guide pratique de l'Oric-Atmos — Michel Bussac/Robert Lagoutte

Des programmes pour votre Oric-Atmos — Michel Piot

Premiers pas avec le Commodore 64 — Ian Stewart/Robin Jones

Initiation à LOGO — Doris Avram/Michèle Weidenfeld

LOGO, Manuel de référence — Doris Avram/Tristan Savatier/Michèle Weidenfeld et l'équipe de S.O.L.I.

Guide du MO 5 — André Deledicq

Faites vos jeux en assembleur sur TO7 et TO7-70 — Michel Oury

Manuel de l'Assembleur du 6809 et du TO7 et TO7-70 — Michel Weissgerber

Manuel de l'assembleur du 6809 du MO5 — Michel Weissgerber

Initiation au FORTH — S.E.F.I.

Guide pratique du vidéotex et du minitel — Jean-Pierre Saboureau/Geneviève Bouché

Guide pratique de l'enseignement assisté par ordinateur — Jean-Michel Lefèvre

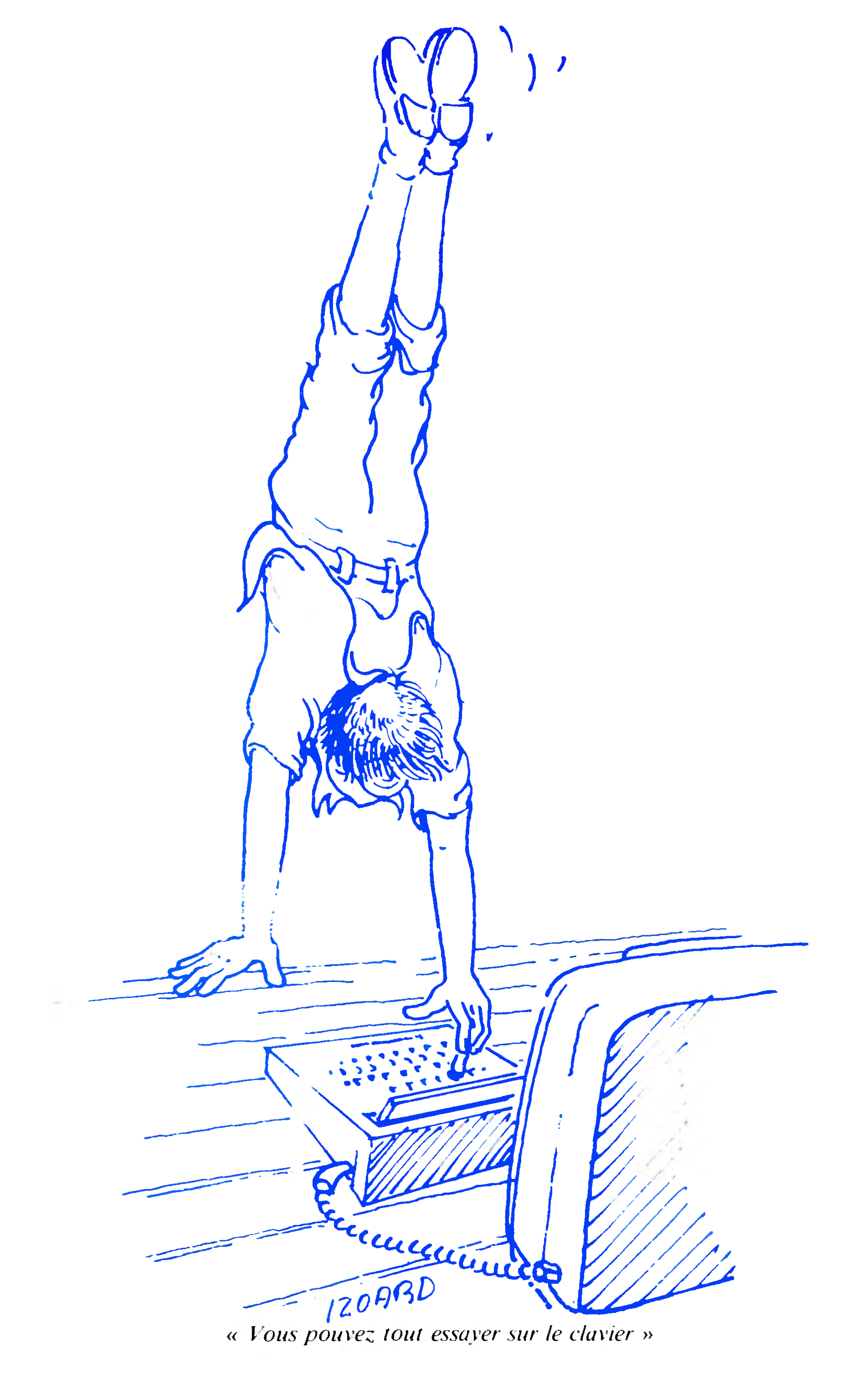
La face cachée du TO7/TO7-70 — Jean-Baptiste Touchard

Manuel technique du TO7/TO7-70 — Michel Oury

Manuel technique du MO5 — Michel Oury

(3[e] édition revue et corrigée)
Ce volume porte la référence
ISBN 2-7124-0510-2

CEDIC, 32, boulevard Saint-Germain, 75005 - PARIS



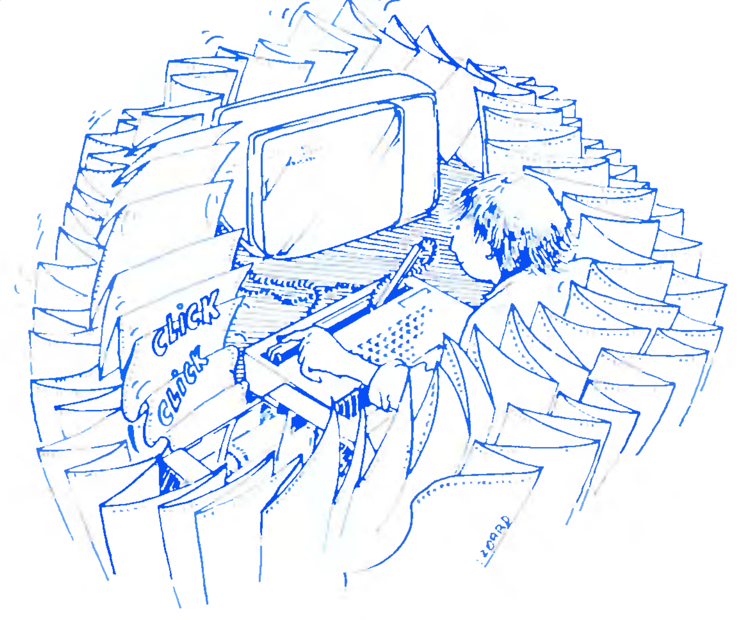
« Vous pouvez tout essayer sur le clavier »

Avant-propos

Vous venez d'acquérir votre premier micro-ordinateur et vous voulez apprendre à le programmer. Le but de cet ouvrage est de vous faciliter le début de cet apprentissage.
Tout d'abord, il faut remarquer qu'apprendre la programmation ne nécessite pas de connaissances préalables bien particulières. Nul besoin d'être scientifique ou féru de mathématiques : dans un premier temps, il suffit des quatre opérations et d'un peu de logique.

Vous constaterez rapidement que le langage BASIC utilisé par votre micro-ordinateur est facile à apprendre. Il a été conçu pour des débutants, comme son nom l'indique : Beginners All purpose Symbolic Instruction Code (langage tous usages pour débutants). Le noyau du langage, ce qui est essentiel pour écrire un programme, est très réduit. Les instructions et les détails supplémentaires qui vous seront bien utiles par la suite, ne sont pas du tout indispensables au départ.

Nous avons introduit les différentes notions qui interviennent dans la programmation, avec les principales instructions du langage, de la manière la plus naturelle possible. Un langage est un outil qui s'élabore, s'améliore, s'affine au fur et à mesure des besoins. Nous avons tenté de mettre en évidence cette construction des différentes instructions à partir des besoins de la programmation.



Il n'est pas possible d'apprendre un langage et de se l'approprier vraiment sans écrire beaucoup de programmes. L'exposé des instructions principales de BASIC se fait au cours des chapitres à partir d'exemples ou d'idées de programmes. Ces exemples vous permettront, en même temps que le langage, de découvrir les particularités de votre micro-ordinateur dans les domaines graphiques et sonores.

Ne lisez pas cet ouvrage d'un seul trait. Etudiez les programmes et les instructions chapitre par chapitre. A partir d'une idée, essayez vos propres solutions, au besoin en allant chercher des informations dans le manuel de référence. Puis revenez à la suite. Vous y apprendrez l'essentiel de BASIC, toutes les instructions importantes, toutes celles qui utilisent pleinement la machine.
Cependant toutes les possibilités du BASIC du TO7 et du TO7-70 ne sont pas mentionnées. S'il vous manque quelque détail particulier, allez le puiser dans le manuel de référence. C'est une habitude qu'il vous faudra prendre petit à petit.
Notre souhait est que cet ouvrage vous permette de devenir autonome.

La deuxième partie comporte des sujets de problèmes et vous propose de les résoudre. C'est le meilleur moyen d'apprendre la programmation.
Essayez de les résoudre vous-même avant de regarder la solution que nous en avons donnée. La vôtre sera peut-être différente, mais si elle est correcte, c'est le principal.

Lorsque vous aurez écrit un certain nombre de programmes, vous vous rendrez compte qu'il faut connaître les instructions mais aussi savoir choisir la méthode convenant à tel problème, que la "lisibilité" d'un programme est importante, qu'écrire un programme sans fautes demande toujours une réflexion préalable avec un papier et un crayon, qu'il existe des problèmes fréquents dont il est utile de bien connaître la solution, etc.

Arrivé à ce stade, vous aurez franchi un grand pas. Vous serez en mesure de programmer grand nombre de jeux, d'exercices, de calculs, de dialogues.

Notre objectif s'arrête là.

A vos machines.

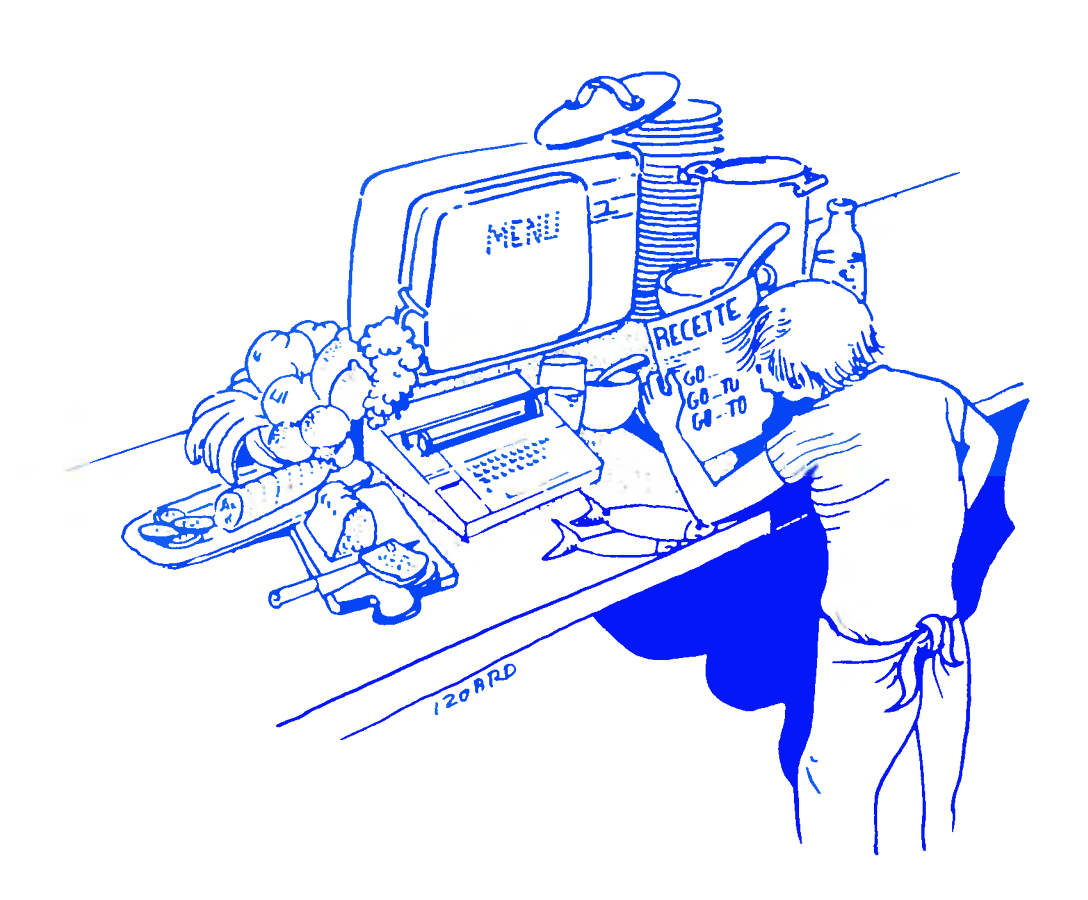
Sommaire

Annexes

Chapitre I/Premier contact avec l'ordinateur

Tout est branché. Votre téléviseur est allumé.

Sur l'écran s'affiche le "menu" que vous proposera l'ordinateur à chaque fois que vous le brancherez.



Un ordinateur ne comprend qu'un langage codé. Comme la plupart des micro-ordinateurs, le TO7 utilise comme langage le BASIC. Vous savez que d'autres langages sont utilisés en informatique suivant les applications (calculs, gestion ...) : Fortran, Cobol, Pascal ...
Le Basic, c'est une chance pour nous, est le plus simple.

Pour pouvoir dialoguer en Basic avec l'ordinateur, il suffit d'appuyer sur le chiffre 1 du clavier (en haut à gauche).
Aussitôt s'affichent sur l'écran le copyright et la version du Basic.

Voyez-vous le petit trait qui clignote sous OK ? C'est le curseur.

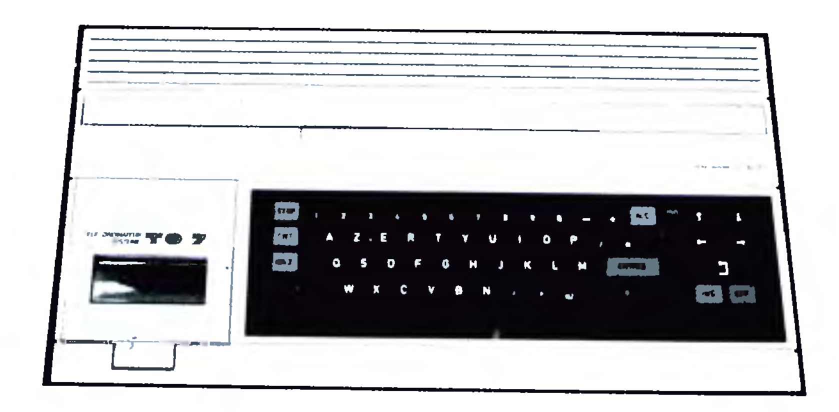
Ainsi l'ordinateur vous dit qu'il est prêt à recevoir vos instructions. Ces instructions, vous allez les lui transmettre par l'intermédiaire du clavier.

Le clavier

C'est l'élément essentiel pour dialoguer avec l'ordinateur. Nous verrons un peu plus loin qu'on peut aussi utiliser le crayon optique.
La première chose à savoir est que vous pouvez tout essayer sur le clavier : l'ordinateur ne risque rien.

Alors allez-y !

Comme toute machine à écrire le clavier comporte les lettres, les chiffres, des signes de ponctuation et des signes mathématiques, ainsi qu'une barre d'espacement. Notez seulement la différence entre la lettre O et le chiffre Ø.



Testez toutes ces touches. Vous voyez qu'après chaque frappe le curseur se déplace d'un cran, indiquant la position où va s'afficher le prochain caractère. En même temps, un petit "bip" sonore, réglable avec le bouton "son" de votre téléviseur, signale à votre entourage que l'ordinateur est au travail.

Les signes jaunes(< > "$...) sont obtenus en appuyant simultanément sur la touche correspondante et sur une des deux touches [•] qui correspondent aux touches "majuscules" d'une machine à écrire.

Vous pouvez encore écrire en lettres minuscules : appuyez simultanément sur une des deux touches [•] et sur la barre d'espacement. Un point rouge s'allume sous "min".

Appuyez alors sur les touches A, Z, E, R... les lettres s'affichent en minuscules sur l'écran.

Si vous voulez repasser en mode majuscule, appuyez à nouveau simultanément sur une touche [•] et sur la barre d'espacement : le point rouge s'éteint et les lettres s'affichent à nouveau en majuscules.

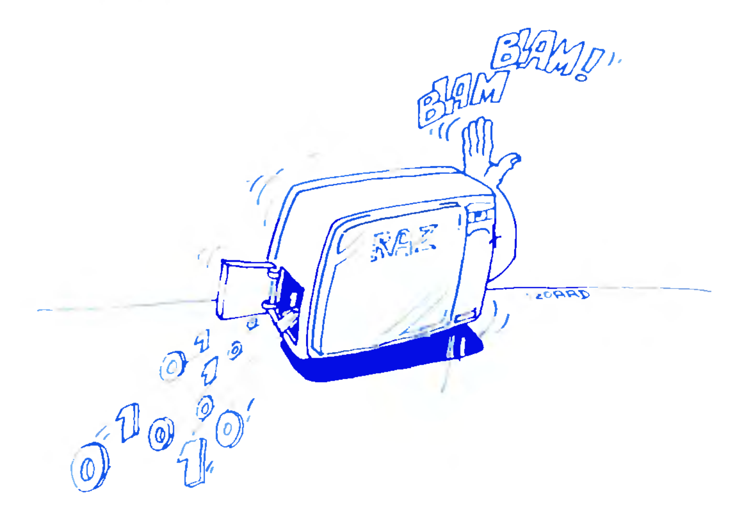
Au cours de ces essais, vous avez sûrement constaté qu'en maintenant la pression sur une touche, le caractère se répète automatiquement à l'écran.

Ceci est bien pratique, en particulier pour créer des espaces.

Maintenant l'écran doit être assez rempli !

Voulez-vous tout effacer ?

C'est le moment d'utiliser la touche [RAZ] (Remise A Zero) : l'écran se vide instantanément et le curseur reprend sa position au début de la première ligne.



Dans la partie droite du clavier vous voyez quatre flèches :

| ↑ | ↓ |
| ← | → |

Elles permettent de déplacer le curseur dans les quatres directions et d'écrire en n'importe quel point de l'écran.

Quant à la touche [⮌] elle ramène automatiquement le curseur en haut à gauche de l'écran.

L'ordinateur est donc une machine à écrire très perfectionnée. Mais heureusement, il est plus que cela : sur une machine à écrire, il n'y a pas de touche [ENTREE].

La touche [ENTREE]

Si cette touche est de couleur différente, c'est que son rôle est aussi différent de celui des autres touches. C'est elle qui donne l'ordre à l'ordinateur de lire (et d'essayer de comprendre ...) ce qui vient d'être écrit à l'écran.

Commençons le dialogue :

[RAZ] — l'écran se vide

[ENTREE] — le curseur se place au début de la ligne suivante : l'ordinateur a rapidement lu la première ligne (qui est vide) et ne fait aucun commentaire !

1 puis [ENTREE] — le chiffre 1 a été lu puisque le curseur s'est placé au début de la ligne suivante, mais l'ordinateur ne fait aucune réponse !

A puis [ENTREE] — Tiens ! L'ordinateur répond :
?SN ERROR
En clair le point d'interrogation signifie qu'il ne vous a pas compris et SN ERROR pour vous dire que vous avez fait une erreur.

Si vous ne le saviez déjà, vous voyez que le vocabulaire de Basic est emprunté à la langue anglaise. Mais rassurez-vous ce vocabulaire ne comporte qu'une trentaine de mots simples. Apparem-

ment la lettre "A" ne fait pas partie de ces mots. Par contre le chiffre 1 semble avoir été accepté.

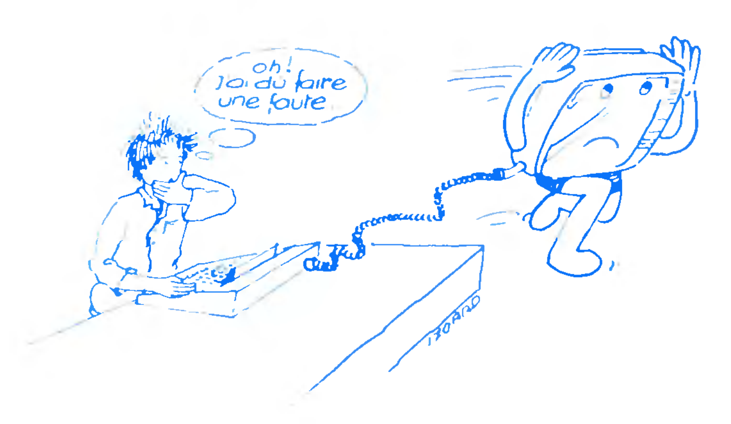
Vous pouvez tester les mots d'anglais que vous connaissez. Il y a de grandes chances que l'ordinateur vous réponde ?SN ERROR inlassablement.

Des goûts et des couleurs

Cela ne se discute pas ! Vous allez donc pouvoir changer les couleurs de l'écran et des caractères avec votre premier mot de Basic : SCREEN, qui en anglais signifie "écran".

Tapez :
SCREEN Ø, 1, 2 puis ENTREE .

Les caractères deviennent noirs, le fond rouge et le pourtour de l'écran vert. Si vous n'obtenez pas cela, vérifiez que vous n'avez pas fait une erreur : virgule oubliée ou faute d'orthographe. L'ordinateur est très pointilleux sur ce chapitre.



Ces couleurs ne vous conviennent pas ? Vous disposez de 8 couleurs différentes.

Pour l'ordinateur ces couleurs sont représentées par des nombres :

Ø pour le noir
1 pour le rouge
2 pour le vert
3 pour le jaune
4 pour le bleu
5 pour le violet (ou magenta)
6 pour le bleu clair (ou cyan)
7 pour le blanc

Le mot SCREEN est une *instruction* de Basic, c'est-à-dire un ordre que l'ordinateur est capable de comprendre et d'exécuter. Cette instruction permet de choisir la couleur des caractères, celle du fond de l'écran et celle du pourtour.

Elle se présente toujours de la même manière : SCREEN, un espace, le chiffre qui donne la couleur des caractères puis séparé par une virgule, le chiffre qui donne la couleur du fond, une virgule et enfin le troisième chiffre qui donne la couleur du pourtour.

Vous pouvez maintenant choisir les couleurs qui vous plaisent : par exemple

SCREEN Ø, 6, 4 puis ENTREE

SCREEN 7, Ø, Ø puis ENTREE

... Il y a 512 possibilités différentes !

Essayez encore
SCREEN 7, 7, Ø.
N'oubliez pas de taper ensuite ENTREE (Ceci doit devenir un réflexe)

Surprise ! Les caractères ont disparu, l'écran est devenu vierge.

Vous venez de redécouvrir l'encre sympathique : en effet les caractères étant de la même couleur que le fond, il est impossible de les distinguer.

Si vous tapez sur une touche, vous entendez le "bip" habituel mais le caractère reste invisible. Pour revenir à une situation plus normale il vous faut taper une nouvelle instruction SCREEN dans laquelle le premier chiffre soit différent du deuxième. Par exemple
SCREEN 7, Ø, Ø.

Il est évidemment difficile de vérifier l'absence de fautes ! Si après plusieurs essais l'écran reste vide, il vous reste un recours : appuyez sur le bouton d'initialisation. Tout reprend à partir du début, vous pouvez recommencer.

Chapitre II/Calculer et écrire

Commençons par les cinq opérations

Un ordinateur, c'est d'abord une calculatrice. Certes, il est plus encombrant mais sa puissance de calcul est aussi beaucoup plus grande ! Mais commençons par le plus simple : comment obtenir la somme 2 + 2 ?

Tapez :
PRINT 2 + 2   puis appuyez sur la touche ENTREE

Si vous n'avez pas fait de faute de frappe, la réponse 4 s'inscrit sur la ligne en-dessous.

C'est l'*instruction* PRINT qui donne l'ordre d'afficher le résultat de l'addition.

Repérez sur le clavier les signes des 4 opérations :

+ pour l'addition
– pour la soustraction
* pour la multiplication, au lieu de " × " qui pourrait être confondu avec la lettre "x"
/ pour la division

Pour afficher le résultat d'une opération, il suffit de la faire précéder de PRINT :

Tapez :
```
PRINT 1Ø – 3
PRINT 4*9
PRINT 5/2
```

Comme sur une calculatrice, la virgule d'un nombre décimal est remplacée par un point ainsi que vous pouvez le constater sur le résultat 2.5 de l'opération 5/2. En outre un nombre comme Ø,25 s'écrira .25 :

Tapez :
```
PRINT 1/4
```

Sur le clavier il existe une cinquième opération possible. Si vous voulez multiplier le nombre 5 plusieurs fois par lui-même, par exemple quatre fois, vous pouvez taper :
```
PRINT 5*5*5*5
```

Cette opération qu'on appelle élévation à la puissance, ou exponentiation peut être effectuée directement en tapant :

PRINT 5^4

Le signe se trouve dans la partie inférieure droite du clavier et il s'obtient en appuyant simultanément sur la touche [a] et sur la touche [•] . Sur l'écran il est remplacé par une flèche vers le haut.

Par exemple pour calculer 4*4

Tapez :

PRINT 4^2

pour 4*4*4 tapez :

PRINT 4^3

pour 4*4*4*4 tapez :

PRINT 4^4

Des opérations en chaîne



Si vous voulez faire faire à l'ordinateur une suite d'additions et de soustractions, par exemple :

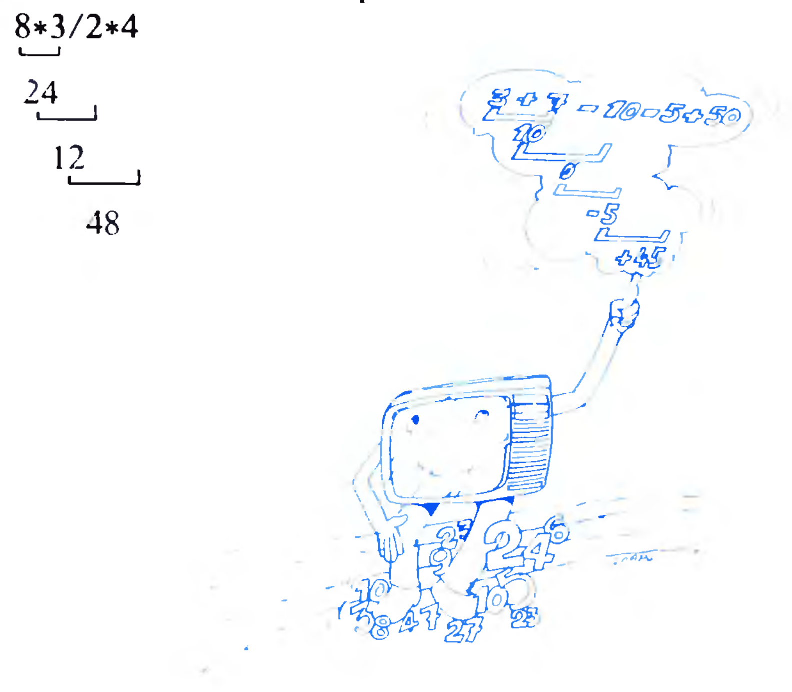
3 + 7 − 1Ø − 5 + 5Ø

l'ordinateur fera comme vous, il effectuera les opérations les unes à la suite des autres à partir de la gauche :

```
3 + 7 − 1Ø − 5 + 5Ø
└───┘
 1Ø
 └──────┘
    Ø
    └───────┘
       − 5
         └───────┘
             + 45
```

De même pour une suite de multiplications et de divisions :

```
8*3/2*4
└─┘
24 ──┘
  12 ───┘
     48
```



Mais si vous mélangez des additions (ou des soustractions) d'une part avec des multiplications (ou des divisions) d'autre part, il n'en est plus ainsi : l'ordinateur ne calcule plus systématiquement de gauche à droite. Les multiplications (et divisions) ont priorité sur les additions (et soustractions) :

```
5 + 3/2
    └─┘
└─ 1.5
 6.5
```

la division a priorité sur l'addition

```
6 - 2*5
    └─┘
└─  1Ø
 - 4
```

la multiplication a priorité sur la soustraction

S'il y a plusieurs multiplications (ou divisions), l'ordinateur les effectue les unes à la suite des autres en commençant par la gauche, puis il effectue les additions (ou soustractions) :

```
6*3 - 15/5*3
└─┘   └──┘
 18     3 ──┘
  └──────9
      9
```

Alors attention à la hiérarchie ! Essayez de raisonner comme l'ordinateur en calculant les expressions suivantes :
5 + 6Ø/2*3 – 8*4
6 – 18/3 + 15*5 – 2

Vérifiez votre résultat en tapant :
PRINT suivi de l'expression à calculer.

Si dans votre calcul apparaissent des élévations à une puissance, alors ce sont elles qui sont calculées en premier. Elles ont encore priorité sur les multiplications et les divisions :

```
5 + 3*4^2
      └─┘
      16
    └──┘
     48
└────┘
 53
```

L'ordre de priorité des opérations est donc le suivant :
— les élévations à une puissance
— les multiplications et les divisions
— les additions et les soustractions

Des parenthèses pour bousculer la hiérarchie

Supposons que vous ayez envie de calculer la moyenne mensuelle de vos dépenses relatives à votre automobile (ou votre moto ou votre bicyclette...). Il faut ajouter toutes les dépenses faites en une année, par exemple 5ØØ F, 2ØØ F, 27Ø F, et diviser le total par 12.

Si vous faites :
PRINT 5ØØ + 2ØØ + 27Ø/12
722.5
est la réponse de l'ordinateur.

Est-ce le résultat cherché ?
Non ! A cause de ces règles de priorités entre opérations, l'ordinateur calcule d'abord la division 27Ø/12 = 22.5 puis il ajoute 2ØØ et 5ØØ.
Résultat : 722.5

Pour contraindre l'ordinateur à calculer d'abord l'addition, il faut la placer entre parenthèses :
PRINT (5ØØ + 2ØØ + 27Ø)/12
85

Grâce aux parenthèses, vous imposez ainsi l'ordre que vous voulez.

Si vous écrivez :
PRINT 66/(3*(7+4))+3
l'ordinateur calcule d'abord la petite parenthèse, celle qui est à l'intérieur de l'autre, puis la deuxième parenthèse, et enfin les opérations qui restent, en suivant les règles de priorité :
(7+4) = 11
3*11 = 33
66/33 = 2
2+3 = 5

En cas de doute, mettez plutôt une paire de parenthèses inutile, cela ne prend guère de place en mémoire et c'est plus sûr ...

Faites des essais de votre crû en comparant le résultat que vous obtenez sur le papier avec celui de l'ordinateur.

Nombres

Au cours des essais précédents, vous êtes peut-être tombé sur des divisions du type 1/3 = Ø.333333333..., c'est-à-dire dont le résultat comporte une suite infinie de chiffres.

Essayez :
PRINT 1/3
.333333
L'ordinateur ne garde donc que 6 chiffres.

Essayez :
PRINT 1/6
.666667

La valeur exacte est Ø.6666666666... Le dernier 6 a donc été remplacé par un 7 dans le résultat arrondi de l'ordinateur.

Une précision de 6 chiffres c'est déjà pas mal, mais si cela ne vous suffit pas sachez qu'il est possible d'obtenir une précision plus grande (voyez le manuel de référence BASIC)

Il faut encore vous prévenir d'une surprise qui peut vous arriver lorsque vous faites des multiplications :

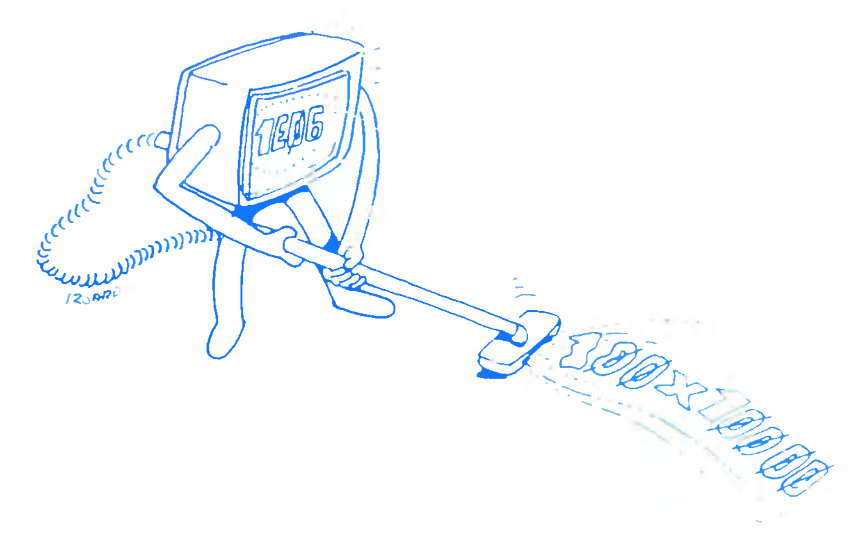
Essayez :
PRINT 1ØØ*1ØØØØ

Certes, c'est prendre un marteau pour écraser une mouche que d'utiliser un ordinateur pour faire une opération aussi simple !

Mais au lieu d'afficher 1ØØØØØØ, il donne comme résultat : 1E+Ø6.

Cette réponse est-elle un code secret ?

Le code est la notation scientifique, utilisée par les calculatrices et les ordinateurs pour écrire les grands nombres :



1ØØØØØØ peut s'écrire $1Ø^6$, soit "1Ø puissance 6". Le nombre 6 est l'exposant, c'est-à-dire le nombre de Ø après le 1. L'ordinateur écrit seulement la valeur de l'exposant, soit la puissance de 1Ø : 1E+Ø6.

523ØØØØ peut aussi s'écrire 5,23 $1Ø^6$, l'ordinateur affichera : 5.23E+Ø6.

Et si vous voulez écrire des mots

L'instruction PRINT ne sert pas qu'à imprimer les résultats des opérations. Elle permet aussi d'imprimer du texte, à condition de le mettre entre guillemets :

```
PRINT "BONJOUR"
PRINT "MARDI 8 MAI"
PRINT "* VITESSE DE LA FUSEE *"
```

Remarquez que l'espace (ou "blanc") est un caractère au même titre que les autres.

L'exigence de l'ordinateur est rigoureuse vis-à-vis des guillemets, sauf pour les nombres. Comparez les résultats de ces deux instructions :

```
PRINT "423Ø"
PRINT  423Ø
```

Puis : PRINT "VITESSE"
PRINT VITESSE
Puis : PRINT "5+7"
PRINT 5+7

En conclusion, tous les caractères qui sont entre guillemets sont imprimés tels quels.

Essayez maintenant :

```
PRINT "5+7 =";5+7
```

Voyez-vous le point-virgule entre les deux expressions? Il demande que l'affichage du résultat de l'opération (5+7 sans guillemets) s'affiche à la suite du texte "5+7 =".

Le rôle du point-virgule vous semblera évident lorsque vous aurez tapé :

```
PRINT "A";"B";"C"
```

Pour les nombres l'ordinateur ajoute un espace devant le nombre, pour un éventuel signe −, et aussi un espace derrière le nombre :

```
PRINT 1;2;3
PRINT -1;-2;-3
```

Corriger ses fautes

Faire des fautes de frappe, cela arrive à tout le monde. Heureusement cela a été prévu ! C'est là que vous allez apprécier les touches avec les petites flèches.

Si par exemple, au lieu de taper :

```
PRINT 1;2;3
```

vous tapez :

```
PRINY 1;2;3
```

Avant d'appuyez sur la touche [ENTREE], ramenez le curseur sous la lettre Y avec la touche [←], puis tapez la lettre T. La correction est faite : le Y a été remplacé par un T.

Vous pouvez alors appuyez sur la touche [ENTREE]. En effet il n'est pas nécessaire que le curseur soit au bout de la ligne pour que celle-ci soit enregistrée. Vérifiez-le : la ligne est lue par l'ordinateur et l'instruction exécutée quelle que soit la position du curseur.

Si vous répétez un caractère involontairement :

```
PPRINT "A";"B"
```

Ramenez le curseur sous le deuxième P, celui qui est en trop, avec la flèche [←].

Vous pouvez effacer cette lettre en appuyant sur la touche [EFF].

Ça y est ?

Vous remarquez qu'en même temps la fin de la ligne se décale d'un cran vers la gauche. Ainsi l'effacement d'une lettre ne provoque pas de "trou".

Si vous appuyez encore sur cette touche [EFF] vous voyez s'effacer sucessivement les lettres R, puis I, puis N.

Arrêtez-vous alors !
PT "A";"B"
(Le curseur est sous le T)

La touche [INS] permet de réintroduire les lettres manquantes : chaque appui sur cette touche insère un blanc devant le curseur, c'est-à-dire ici devant le T.

Il manque trois lettres, il faut donc créer trois blancs entre le P et le T. Appuyez trois fois sur la touche [INS].
Si vous en ouvrez trop, il suffit de les supprimer avec la touche [EFF].
Sur les espaces ainsi créés, ramenez le curseur pour taper les lettres manquantes RIN, et enfin tapez [ENTREE].
Vous voyez que c'est plus long à dire qu'à faire !

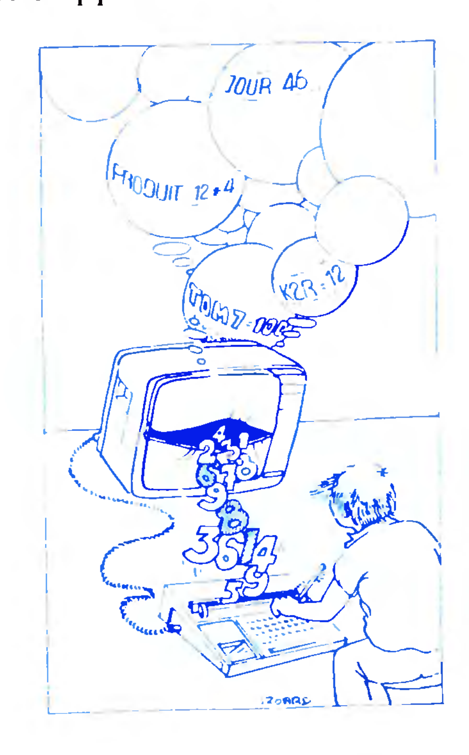
Testez ces touches : [←], [→], [INS] et [EFF], jusqu'à ce que leur usage vous semble naturel pour modifier des caractères, en supprimer ou en rajouter.

Chapitre III/Les variables

Comment utiliser sa mémoire ?

Il ne s'agit pas de la vôtre bien sûr, mais de celle de l'ordinateur. Si vous tapez PRINT "JOUR", l'ordinateur se comporte comme une machine à écrire et affiche ce qui est entre guillemets, c'est-à-dire JOUR.
Si vous voulez qu'il utilise sa mémoire pour y conserver le numéro du jour dans l'année, il faut d'abord lui donner le nombre correspondant. Par exemple pour le 15 février qui est le 46e jour, tapez :
JOUR = 46 puis ENTREE
Vous pouvez vérifier que l'ordinateur sait maintenant cette valeur en tapant :
PRINT JOUR puis ENTREE
Attention ! Pas de guillemets cette fois.
Sur la ligne suivante s'affiche le nombre 46.

L'*instruction* JOUR = 46 réserve une zone de mémoire dans l'ordinateur. Cette zone est repérée par le nom qu'on lui donne : JOUR, et elle reçoit la valeur 46. Une zone de mémoire à laquelle on donne un nom et dans laquelle on peut inscrire une valeur numérique s'appelle une *variable*.



Si vous effacez l'écran avec [RAZ], la valeur reste en mémoire : tapez à nouveau :
PRINT JOUR,
le nombre 46 s'affiche à nouveau sur la ligne suivante.

Vous pourrez ainsi garder en mémoire tous les nombres que vous voulez : il suffit de donner un nom différent à chacun. Les nombres à conserver peuvent être des résultats de calculs :
PRODUIT = 12*4 puis [ENTREE]
PRINT PRODUIT puis [ENTREE]
La réponse s'affiche directement : 48.

Remarquez que cette fois nous n'avons pas mis de "blancs" de chaque côté du signe = . Ils sont facultatifs, comme dans beaucoup d'autres instructions. Par contre, il faut toujours mettre le nom de la variable à gauche de ce signe = . L'instruction peut se lire :
"La variable appelée PRODUIT *reçoit* la valeur 12*4"

Si, comme M. Jourdain, vous essayez d'inverser le sujet et le complément dans la phrase précédente, celle-ci perd tout son sens.

Changer la valeur d'une variable

Pour changer la valeur d'une variable, il suffit de lui affecter une nouvelle valeur :
PRODUIT = 1ØØ
PRINT PRODUIT
(Vous devez avoir maintenant le réflexe de taper [ENTREE] après chaque instruction.)
La réponse est 1ØØ : l'ancienne valeur 48 est effacée, sans espoir de retour.

Essayez maintenant :
PRINT SOMME
Vous obtenez Ø. Cela n'est pas étonnant : vous n'avez encore affecté aucune valeur à cette variable. Il en est ainsi pour tous les noms de variables sauf pour JOUR et PRODUIT.

Le rôle des guillemets dans l'instruction PRINT est donc très important :
PRINT "A" signifie : afficher la *lettre* A
PRINT A signifie : afficher la valeur numérique de la *variable* qui s'appelle A

Noms de variables

Vous pouvez essayez PRINT suivi de tous les noms possibles et imaginables de variables. Il y a de grandes chances que vous obteniez toujours Ø.

Au lieu de taper PRINT, vous pouvez utiliser le point d'interrogation :

?A est équivalent à PRINT A

Cependant si vous tapez :

?SCREENA (sans oublier de faire ensuite ENTREE)

vous obtenez le message d'erreur ?SN ERROR.

En effet, un nom de variable ne doit pas commencer par un mot du langage BASIC ; ici le nom de variable contient l'instruction SCREEN. De même l'instruction NOTE = 18

sera refusée car NOT fait partie des mots utilisés par le BASIC. La liste de ces mots est donnée dans l'annexe 5.

Par contre vous pouvez utiliser des chiffres dans la définition d'une variable :

K25 = 12

TØ7 = 1ØØØ

Cependant le 1er caractère doit toujours être une lettre.



Entrez maintenant les deux variables suivantes :
VARIABLEDE22CARACTERES = 1ØØ
VARIABLEDE22CARA = 2ØØ
(Ne mettez pas de "blancs" entre VARIABLE et DE)

Demandez ensuite :
PRINT VARIABLEDE22CARACTERES
Vous attendez 1ØØ comme réponse et vous obtenez 2ØØ !

Que s'est-il passé ? Dans un nom de variable, l'ordinateur ne retient que les *seize* premiers caractères. Or les deux variables que vous avez fastidieusement rentrées dans l'ordinateur ont les mêmes seize premiers caractères ; elles définissent donc une seule et même variable. La valeur conservée en mémoire est la dernière rentrée, soit 2ØØ.

Une variable, ça peut toujours servir

Lorsque vous avez défini une variable, vous pouvez l'utiliser dans des calculs ou des instructions :
SOMME = 12+4
PRINT SOMME +11
PRINT SOMME *1Ø
PRINT SOMME /4
La valeur de la variable SOMME n'est pas modifiée par ces calculs. Vérifiez-le.

Une nouvelle variable peut même être définie à partir de cette variable SOMME :
MOYENNE = SOMME/2
PRINT MOYENNE

Remarquez encore que la variable que l'on appelle MOYENNE se trouve à gauche du signe = : elle reçoit la valeur numérique SOMME/2.

Tout ce qui peut être représenté par un nombre peut devenir une variable, par exemple une couleur :
C = 2
SCREEN 7, C, C
SCREEN 7, C, C+2

Attention au signe =

Il faut bien voir que le signe = ne représente pas ici une égalité

comme en mathématiques, mais une *instruction*. Cette instruction affecte à la *variable* dont le nom est à gauche du signe = la *valeur* de ce qui est à droite de ce signe = .

Supposons qu'aujourd'hui le prix de la baguette de pain soit 2ØØ (nous ne précisons pas les unités !) ; on peut définir la variable PRIX :

```
PRIX = 2ØØ
```

Il est alors facile de calculer le prix de 4 baguettes, 3 baguettes 3/4...

Mais au bout d'un certain temps, les prix ont augmenté et en particulier la baguette a augmenté de 1Ø unités. Le nouveau prix est égal à l'ancien augmenté de 1Ø, d'où la nouvelle instruction :

```
PRIX = PRIX+1Ø
```

Cette ligne implique ici PRIX = 21Ø.

Vous pouvez le vérifier en tapant :

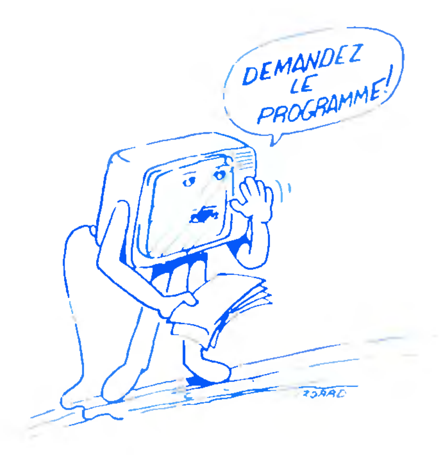
```
PRINT PRIX
```

Vous savez maintenant comment introduire une variable pour conserver en mémoire des données numériques. Vous pouvez entrer tous les prix qui vous intéressent.

Une petite remarque : cette mémoire n'est pas immortelle. En particulier, si vous éteignez l'ordinateur le soir, ne lui demandez pas de vous redonner le lendemain matin le prix du kilo de carottes, il ne s'en souvient plus !

Nous verrons plus loin comment il est possible de conserver des informations avec le lecteur-enregistreur de programmes à cassettes.

Chapitre IV/Comment faire un programme



Faire les choses dans l'ordre ?

Nous avons vu que l'ordinateur peut garder en mémoire autant de nombres que vous voulez : il suffit de leur donner des noms. Mais il peut encore beaucoup plus : garder en mémoire des centaines d'instructions différentes.

Pour que ces instructions s'effectuent dans un ordre déterminé, il faut préciser cet ordre à l'ordinateur par des numéros.

Tapez :

RAZ pour vider l'écran,	puis :	
1 AGE = 18	puis	ENTREE
2 PRINT "AGE EN MOIS:"	puis	ENTREE
3 PRINT AGE *12	puis	ENTREE

Ces trois lignes constituent un programme, c'est-à-dire une suite d'instructions.
Les chiffres 1, 2 et 3 sont les numéros de lignes qui précisent l'ordre dans lequel doivent être lues les instructions :
La ligne 1 affecte la valeur 18 (sous-entendu : années) à la variable AGE.
La ligne 2 demande l'affichage de l'expression "AGE EN MOIS:".
La ligne 3 calcule puis affiche la valeur du produit AGE *12.

Mais apparemment rien ne se passe.
Lorsque vous avez entré vos 3 lignes, l'ordinateur les a seulement *enregistrées* dans sa mémoire.
Pour qu'il passe à *exécution*, c'est-à-dire qu'il obéisse à la suite des instructions, il faut le lui demander par une nouvelle commande : RUN

Tapez :
RUN puis [ENTREE]
Vous voyez alors s'afficher le résultat de l'action du programme :
AGE EN MOIS:
216

Evidemment si vous avez 18 ans depuis plusieurs mois, ce résultat est approximatif. Il faudrait perfectionner le programme, c'est souvent comme cela en informatique.

Videz l'écran avec [RAZ] et tapez à nouveau :
RUN
Le programme s'exécute une deuxième fois. Vous pouvez le faire tourner autant de fois que vous voulez : les trois instructions sont conservées dans l'ordre en mémoire.

Mais vous, les avez-vous encore en mémoire ?

Pour afficher à nouveau le programme il suffit de taper :
LIST puis, comme toujours, [ENTREE]
Vous pouvez encore le faire tourner avec :
RUN.

Vous voyez que l'affichage du programme à l'écran et son exécution sont deux choses bien distinctes : on peut faire "tourner" le programme sans qu'il soit affiché à l'écran et inversement son affichage à l'écran ne le fait pas tourner.

Modifier le programme

Si vous voulez modifier une ligne avant de l'avoir enregistrée, c'est-à-dire avant d'avoir appuyé sur la touche [ENTREE], il suffit d'utiliser les touches [←], [→] et [INS], [EFF].

La correction se fait de la même manière, que la ligne ait un numéro, (on est alors "en mode programme"), ou qu'elle n'en ait pas (on est alors "en mode direct").
Mais il est aussi possible de modifier une ligne déjà enregistrée : on ramène alors le curseur à l'endroit voulu avec les flèches [↑] et [↓]. Puis on effectue la modification à l'intérieur de la ligne.

Par exemple si nous voulons que le programme affiche :
AGE EN MOIS:
(AU MOINS)216

au lieu de
AGE EN MOIS:
216

Listez le programme :

```
1 AGE=18
2 PRINT"AGE EN MOIS:"
3 PRINT AGE *12
```

Ramenez le curseur sous la ligne 3 après le mot PRINT. Insérez un certain nombre de blancs (s'il y en a trop, vous les effacerez ensuite), puis tapez "(AU MOINS)";
La ligne 3 devient :
3 PRINT "(AU MOINS)";AGE *12

Inutile de ramener le curseur au bout de la ligne, mais par contre n'oubliez pas d'appuyer sur la touche [ENTREE] pour enregistrer la nouvelle version de cette ligne.
Si après avoir effectué la correction, vous ramenez le curseur sur la ligne suivante avec la touche [↓] sans appuyer sur [ENTREE], la nouvelle version de la ligne n'est pas enregistrée. Vérifiez-le.

Vous pouvez aussi ajouter une ligne à la suite du programme : il suffit de lui donner un numéro supérieur à celui de la dernière ligne du programme. Par exemple :
10 PRINT "AU REVOIR"
Listez à nouveau le programme pour vérifier que la ligne a bien été enregistrée.

Vous pouvez même intercaler une nouvelle ligne entre deux lignes du programme en donnant à cette ligne un numéro compris entre les deux autres :
4 PRINT "AGE EN JOURS:"
LIST
et encore :
5 PRINT "(AU MOINS)";AGE*12*30
LIST

D'où le programme final :

```
1 AGE=18
2 PRINT"AGE EN MOIS:"
3 PRINT "(AU MOINS)";AGE*12
4 PRINT "AGE EN JOURS:"
5 PRINT "(AU MOINS)";AGE*12*30
10 PRINT "AU REVOIR"
```

Mode direct et mode programme

Votre programme est en mémoire. Peut-être y reviendrez vous tout à l'heure pour le modifier encore ?

Mais en attendant rien ne vous empêche d'utiliser l'ordinateur en mode direct, c'est-à-dire en lui donnant des instructions à exécuter immédiatement, donc sans numéro de ligne. Par exemple pour faire des calculs :

```
PRINT 12*15
```

Vous pouvez aussi vérifier l'effet d'une instruction :

```
SCREEN Ø, 6, 1
```

Enfin, ceci permet de connaître la valeur d'une variable lorsque le programme a été exécuté :

```
PRINT AGE
18
```

La valeur 18 a bien été gardée en mémoire.

Modifiez maintenant la ligne 1 du programme en remplaçant 18 par 2Ø.

Vérifiez la modification avec :

LIST puis

```
RUN
```

Si vous demandez à nouveau en mode direct :

```
PRINT AGE
2Ø
```

vous obtenez la nouvelle valeur de la variable : 2Ø

Vider la mémoire

Si vous voulez essayer un nouveau programme, il faut d'abord faire subir un lavage de cerveau à votre ordinateur. Une instruction est prévue pour cela : l'instruction NEW qui efface le programme en mémoire.

Tapez :

```
NEW
```

Vérifiez qu'il n'y a plus d'instructions en mémoire en tapant :

```
LIST
```

L'instruction :

```
RUN
```

n'a bien sûr plus aucun effet. Vous pouvez aussi voir quelle est la valeur de la variable AGE en demandant :

PRINT AGE
Ø
La réponse est Ø : l'instruction NEW remet également toutes les variables à Ø.

La machine est donc prête à enregistrer un nouveau programme. Lorsque vous branchez l'ordinateur, il en est de même : la mémoire est vide au départ.

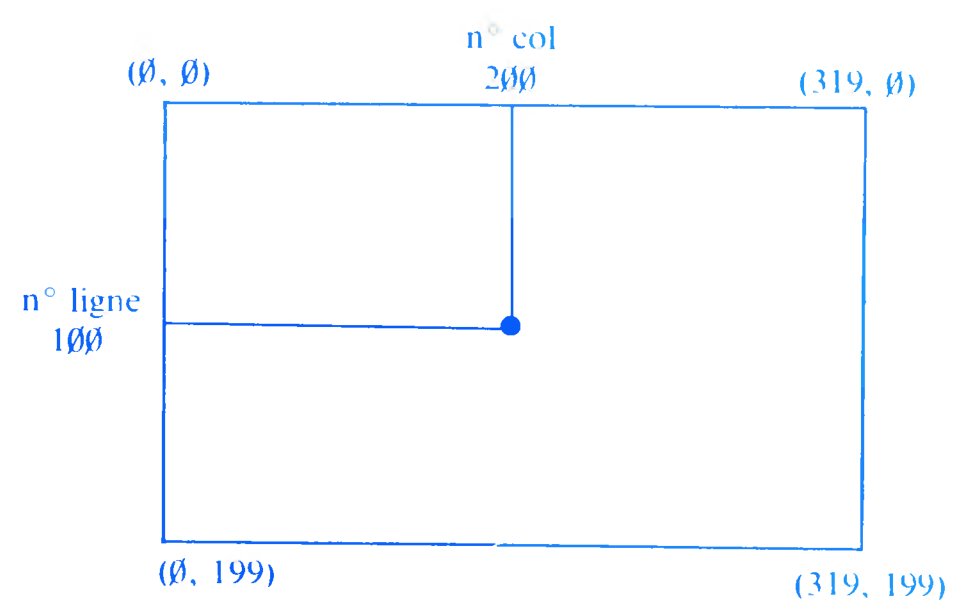
Changement de programme : un peu de dessin

Avec les instructions graphiques du TO7 vous pourrez dessiner à l'écran, animer ces dessins, créer des jeux...

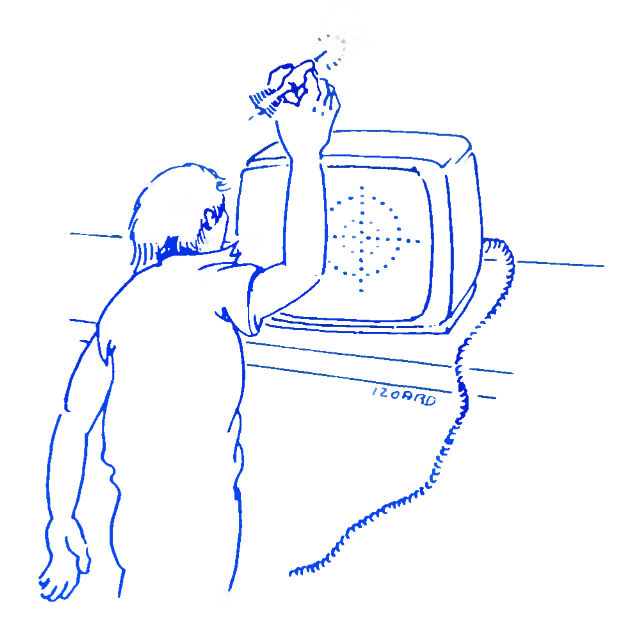
Vous avez déjà vu avec l'instruction SCREEN la partie rectangulaire de l'écran dans laquelle travaille l'ordinateur. Ce rectangle comporte 32Ø points dans le sens de la largeur et 2ØØ points dans le sens de la hauteur, soit 32Ø*2ØØ = 64ØØØ points.

Comment distinguer tous ces points ?

Pour l'ordinateur, chaque point est défini par deux nombres : son numéro de *colonne* et son numéro de *ligne*.
Les colonnes sont numérotées de Ø à 319 à partir de la gauche.
Les lignes sont numérotées de Ø à 199 à partir du haut.



Ainsi (2ØØ, 1ØØ) représente le point à l'intersection de la colonne n° 2ØØ et de la ligne n° 1ØØ. On appellera ces deux nombres ses *coordonnées* (notez bien que les numéros de lignes augmentent de haut en bas).



Vous voudriez le voir ? Il faut le demander de manière compréhensible pour l'ordinateur. L'instruction PSET (pour SET a Point = placer un point) permet d'allumer n'importe quel point à l'écran :

```
1Ø PSET (2ØØ, 1ØØ)
RUN
```

Voyez-vous le point lumineux (2ØØ, 1ØØ) ?
Non ? N'avez-vous pas oublié d'appuyer sur la touche [ENTREE] ?
Peut-être y a-t-il déjà beaucoup de caractères sur votre écran ? Dans ce cas, il n'est pas facile de distinguer un point.

Videz alors l'écran :
[RAZ]
puis faites à nouveau tourner le programme :

```
RUN
```

Cette fois le point ne doit plus vous échapper.

Vous pouvez l'allumer dans la couleur de votre choix : précisez le numéro de la couleur choisie après les coordonnées du point :

```
1Ø PSET (2ØØ, 1ØØ), 1      pour un point rouge
1Ø PSET (2ØØ, 2ØØ), 2      pour un point vert
1Ø PSET (2ØØ, 3ØØ), 3      pour un point jaune
```

etc.

Pour éviter de vider l'écran avant chaque exécution avec la touche [RAZ], on peut prévoir ce nettoyage systématiquement au début de l'exécution du programme avec l'instruction CLS (pour CLEAR SCREEN = nettoyer l'écran).

C'est une bonne habitude que de placer cette instruction au début de chaque programme qui utilise des instructions graphiques comme PSET :

```
5 CLS
LIST
RUN
```

Ainsi, même si votre écran est rempli avant de faire RUN , l'exécution du programme commencera par le vider.

Votre point est un peu perdu, tout seul à l'écran. Si vous allumiez son voisin de droite ? de gauche ? de dessus ? de dessous ? Essayez de former une petite croix autour du premier point. N'oubliez pas que les colonnes sont numérotées à partir de la gauche et les lignes à partir du haut.

Voici la solution :

```
5 CLS
10 PSET (200,100),1
20 PSET (199,100),1     point à gauche
30 PSET (201,100),1     point à droite
40 PSET (200,99),1      point au dessus
50 PSET (200,101),1     point au dessous
```

Que se passe-t-il si dans l'instruction PSET vous donnez des numéros de colonne ou de ligne supérieurs aux valeurs limites (319 pour les colonnes et 199 pour les lignes) ?

Essayez par exemple :

```
PSET (400, 100)
```

Vous voyez le point au bout de la ligne 100 à droite. L'effet est donc le même que si vous aviez tapé :

```
PSET (319, 100)
```

De même pour les lignes :

```
PSET (200, 300)
```

Le point est en bas de la colonne 200 et cette instruction est donc équivalente à :

```
PSET (200, 199)
```

Vous pouvez maintenant dessiner des points comme vous voulez à l'écran. Mais nous allons voir de nouvelles instructions qui s'ajouteront à l'instruction PSET et permettront de dessiner plus rapidement. En effet un dessin point par point suppose une ligne de programme par point.

Chapitre V/Des boucles

Une ligne en deux lignes

Avez-vous déjà vu un ordinateur qui ne sait plus s'arrêter ? C'est très simple : il suffit de lui faire répéter indéfiniment les mêmes instructions.

Comme un programme de quelques instructions est exécuté en moins d'un millième de seconde, vous voyez que cela peut aller loin.

Si vous avez un programme en mémoire, tapez :

```
NEW
```

```
10 PRINT "JE ME REPETE, MAIS JE NE SUIS
PAS FOU."
```

Cette ligne de programme est plus longue qu'une ligne à l'écran : après le mot "SUIS", le curseur revient au début de la ligne suivante. Ne tapez pas sur la touche [ENTREE]. Continuez à taper les derniers mots et enfin tapez sur la touche [ENTREE].

Une "ligne" de programme, c'est la suite de tous les caractères (y compris les espaces) entre le numéro de ligne et l'appui sur la touche [ENTREE].
Sur le TO7, une ligne de programme peut contenir jusqu'à 255 caractères, c'est-à-dire environ 6 lignes et demi à l'écran. En effet, une ligne à l'écran comporte 4∅ caractères.

Faites tourner le programme. La phrase "JE ME REPETE, MAIS JE NE SUIS PAS FOU" s'affiche sur une seule ligne à l'écran.
En général, on facilite la lecture des programmes en limitant la longueur d'une ligne de programme à une ligne à l'écran.
Ecrivons donc ce programme sur deux lignes.

Tapez :

puis, avec la touche [EFF], effacez la fin de la ligne à partir de la 1re lettre "M" du mot "MAIS".

Vous voyez les caractères de la deuxième ligne "remonter" à la fin de la première avant d'être effacés.
Terminez cette ligne par des guillemets puis un point-virgule, n'oubliez pas de taper sur [ENTREE], puis tapez la deuxième

ligne, ce qui donne :

```
10 PRINT "JE ME REPETE,";
20 PRINT " MAIS JE NE SUIS PAS FOU."
```

Exécutez le programme avec :
RUN
il donne le même résultat que l'ancien.
C'est le point-virgule à la fin de la ligne 1Ø qui demande l'affichage de l'instruction PRINT suivante à la suite de la première. Vérifiez le en effaçant ce point-virgule et en exécutant à nouveau le programme. Remettez ensuite le point-virgule à sa place.

Commencez-vous à apprécier les petites flèches pour faire les corrections ?

Vous voyez que les modifications de programmes seront très faciles avec cet *éditeur* (c'est ainsi qu'on appelle le dispositif de correction sur un ordinateur).

Qu'est-ce qu'une boucle ?

Pour voir l'ordinateur justifier la phrase affichée à l'écran, nous allons utiliser une nouvelle instruction : GOTO 1Ø qui signifie "aller à la ligne 1Ø".
C'est comme au jeu de l'oie lorsqu'on tombe sur une case qui vous dit : "aller à la case 1Ø". Ce peut être pour reculer ou pour avancer.

Ajoutez donc la ligne 3Ø :

```
10 PRINT "JE ME REPETE,";
20 PRINT " MAIS JE NE SUIS PAS FOU."
30 GOTO 10
```

Imaginez-vous dans quel ordre vont s'exécuter les instructions ?
RUN
Evidemment il se répète pas mal ! Vous voyez que lorsque les 24 lignes de l'écran ont été utilisées, c'est-à-dire lorsque le programme a tourné 24 fois, seule la dernière ligne à l'écran clignote.

A chaque nouveau tour de programme, toutes les lignes sont décalées d'un cran vers le haut et l'affichage se fait sur la dernière.

Comment l'arrêter ?

Vous pouvez toujours aller fermer le compteur de l'appartement, ou débrancher l'ordinateur. Mais ce cas de figure a été prévu : appuyez simultanément sur la touche [CNT] et sur la lettre C . Prenez votre temps : l'ordinateur ne chauffe pas.

Ouf ! Le programme s'arrête.

L'ordinateur vous précise même qu'il s'est arrêté à la ligne 2Ø :

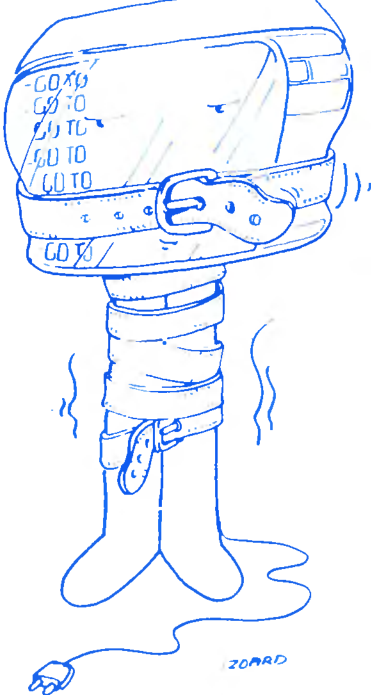
Break in 2Ø (ou peut-être 1Ø ou 3Ø).

Si vous n'arrivez pas à arrêter cette machine infernale, vous pouvez toujours appuyer sur la touche d'initialisation. Cela n'efface pas votre programme dans la mémoire.

Lorsqu'un programme exécute ainsi indéfiniment les mêmes lignes, ici 1Ø, 2Ø, 3Ø, 1Ø, 2Ø, 3Ø, 1Ø... on dit qu'il exécute une boucle. Autrement dit, il tourne en rond. Mais ceci est très souvent utile en programmation.

Si vous n'avez pas débranché l'ordinateur et arrêté le programme avec [CNT] C , vous pouvez faire repartir la boucle en tapant :

CONT



Tourner en rond pour tracer une droite

Vous savez allumer un point à l'écran grâce à l'instruction PSET. Or une droite c'est une suite de points alignés.

Une droite verticale, par exemple la colonne 3ØØ, est formée des 2ØØ points ayant le même numéro de colonne (3ØØ) et des numéros de lignes variant de Ø à 199.

Pour dessiner cette colonne à l'écran, on peut imaginer un programme de 2ØØ instructions PSET :

```
5   CLS
1Ø PSET (3ØØ,Ø)
2Ø PSET (3ØØ, 1)
3Ø PSET (3ØØ, 2)
...................
```

et ainsi de suite jusqu'à :

```
2ØØØ PSET (3ØØ, 199)
```

Si vous avez envie d'essayer quelques lignes de ce "petit" programme, voici le moyen d'éviter de taper des lignes identiques ou presque identiques. Tapez :

```
NEW
```

puis les lignes 5 et 1Ø.
Remontez le curseur sous le numéro de ligne "1Ø" et écrivez un "2" à la place du "1", déplacez ensuite le curseur sous la même ligne jusqu'au chiffre "Ø" qui suit la virgule et changez-le en "1".
En appuyant sur la touche ENTREE, vous créez la ligne 2Ø. La ligne 1Ø n'est pas supprimée pour autant : vérifiez-le avec LIST .
Et ainsi de suite jusqu'à la ligne 2ØØØ si le cœur vous en dit !

En informatique, comme ailleurs, il y a toujours plusieurs moyens d'arriver au but. Mais ils ne sont pas tous aussi bons.

Ici nous faisons recommencer 2ØØ fois la même opération : allumer un point à l'écran. La seule chose qui change est la valeur du numéro de ligne du point à allumer.
Pourquoi ne pas faire de ce numéro de ligne une variable, par exemple LIGNE, et faire boucler l'ordinateur en augmentant à chaque tour la variable d'une unité ?

Regardez ce programme et comparez-le au précédent :

```
5 CLS
10 LIGNE = 0
20    PSET (300,LIGNE)
30    LIGNE = LIGNE + 1
40    GOTO 20
```

La ligne 1Ø introduit la variable LIGNE et lui affecte la valeur Ø. On dit qu'elle "initialise" la variable à Ø.
La ligne 2Ø allume d'abord le point (3ØØ, Ø), c'est-à-dire en haut de l'écran.

La ligne 3Ø peut se lire : "la variable LIGNE reçoit l'ancienne valeur de cette variable, augmentée de 1". A la fin de cette instruction, la variable LIGNE vaut donc 1.
La ligne 4Ø renvoie à la ligne 2Ø qui va alors allumer le point (3ØØ, 1)... et ainsi de suite.

A chaque tour de boucle, la variable LIGNE augmente de 1 et c'est le point juste au-dessous du précédent qui est allumé.
Remarquez que les lignes 2Ø, 3Ø et 4Ø sont décalées vers la droite pour mettre en évidence les 3 instructions de la boucle.
Tapez ce programme et faites le exécuter.

La droite verticale s'affiche à l'écran à partir du haut.
Lorsque l'opération est terminée, essayez d'appuyer sur une touche du clavier. Vous entendez le BIP habituel mais rien ne s'affiche à l'écran : le programme tourne toujours et l'ordinateur ne peut pas écouter de nouvelles commandes.

En effet lorsque la variable LIGNE atteint la valeur 199 la droite est terminée ; au tour suivant LIGNE vaut 2ØØ et l'instruction PSET allume toujours le même point (3ØØ, 199).

Pour arrêter un programme, un seul moyen : taper simultanément sur [CNT] et la lettre C.
Ça y est ?

Un programme peut donc boucler sans que cela se voie. Bien que la consommation électrique de l'ordinateur soit assez faible ceci est naturellement à éviter : lorsqu'on prévoit une boucle il faut aussi prévoir comment la terminer.

Vous pouvez voir ici combien l'ordinateur a fait de tours pour rien en tapant en mode direct (c'est-à-dire sans numéro de ligne) :
PRINT LIGNE
Vous obtenez ainsi la valeur de la variable LIGNE au dernier tour effectué avant l'arrêt du programme par [CNT] C.

Arrêter et reprendre

Il y a une touche sur le clavier qui n'a pas encore été utilisée : la touche [STOP].
Comme [CNT] C, cette touche permet d'arrêter un programme en cours d'exécution. Mais cet arrêt n'est pas définitif : on peut reprendre l'exécution du programme à la ligne où elle s'est arrêtée en appuyant sur une touche quelconque du clavier.

Faites exécuter le programme précédent :
RUN
et appuyez sur :
[STOP]
alors que le spot n'est pas encore arrivé en bas de l'écran : vous le voyez s'arrêter.
Appuyez sur une touche quelconque : il repart du même point et finit de dessiner sa droite.
Pour arrêter définitivement le programme, faites enfin [CNT] C.

Prenez maintenant une feuille de papier pour essayer d'écrire un programme qui tracera une droite horizontale à l'écran en allant de la gauche vers la droite (par exemple la ligne numéro 15Ø).

Voici la solution :

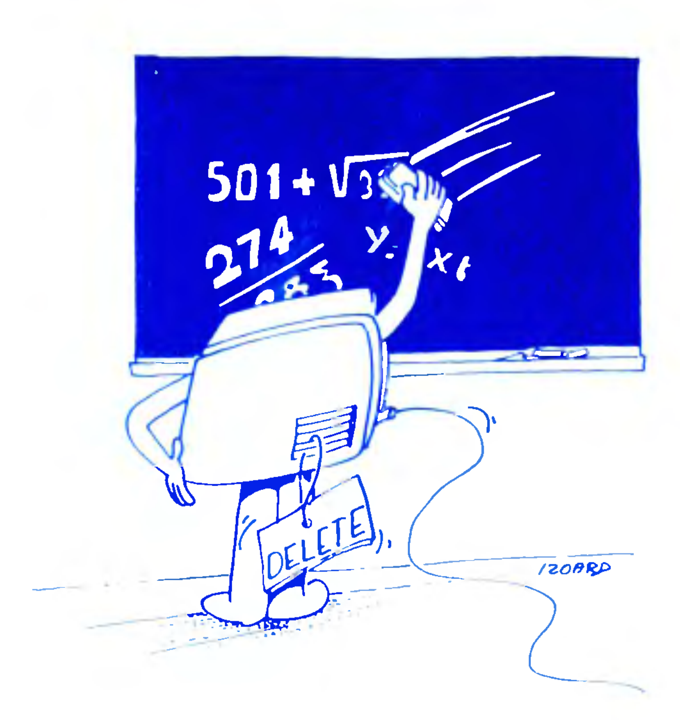
```
5 CLS
10 COLONNE = 0
20    PSET (COLONNE,150)
30    COLONNE = COLONNE + 1
40    GOTO 20
```

Vous remarquez tout de suite qu'il ressemble au précédent : les lignes 5 et 4Ø sont même identiques. Inutile donc d'effacer le précédent par NEW. Nous allons nous contenter de réécrire les lignes qui sont différentes :

```
1Ø COLONNE = Ø
2Ø PSET (COLONNE, 15Ø)
3Ø COLONNE = COLONNE + 1
```

Vérifiez en faisant LIST que les anciennes lignes 1Ø, 2Ø et 3Ø ont bien été modifiées.
Pour remplacer une ligne de programme par une autre, il suffit donc de taper la nouvelle ligne avec l'ancien numéro.

Ainsi pour supprimer une ligne, par exemple la ligne 5, vous pouvez écrire :
5
Cela revient à écrire une nouvelle ligne 5 vide, donc à la supprimer.
Il existe aussi une instruction spéciale pour effacer des lignes : l'instruction DELETE.
DELETE 5 effacera la ligne 5
DELETE 1Ø – 3Ø effacera toutes les lignes comprises entre 1Ø et 3Ø.
Il ne vous reste plus qu'à les retaper !



Remarque : Il n'est pas conseillé d'écrire des commandes (RUN, LIST, ...) ou des instructions avec un écran rempli de dessins. En effet, s'il y a déjà quelque chose sur la ligne où vous écrivez, la commande vous sera refusée. Il est préférable d'appuyer sur | RAZ | puis de taper ses commandes ou ses instructions sur un écran propre.

Chapitre VI/Des petites boîtes dans des grandes boîtes

Encore un moyen de tracer une droite : LINE

Dans le chapitre précédent, le programme qui traçait une droite verticale comportait 5 lignes (au lieu des 2ØØ nécessaires pour allumer chaque point successivement).

Il est possible de faire encore mieux.
Regardez le programme suivant :

```
10 CLS
20 LINE (300,0)-(300,199)
```

L'instruction LINE trace directement la droite entre les deux points (3ØØ, Ø) et (3ØØ, 199). Non seulement c'est plus court à écrire, mais encore l'exécution est beaucoup plus rapide.
Essayez maintenant de tracer la droite horizontale numéro 15Ø.

```
10 CLS
20 LINE (0,150)-(319,150)
```

Comme pour l'instruction PSET, vous pouvez choisir la couleur de la droite en faisant suivre la dernière parenthèse d'une virgule et du numéro de la couleur choisie :

20 LINE (Ø, 15Ø) – (319, 15Ø), 2

Vous voyez tout l'intérêt de cette instruction LINE par rapport au programme qui utilise l'instruction PSET dans une boucle.
Mais ce n'est pas tout : essayez d'imaginer comment tracer une droite oblique à l'écran avec PSET. Il faudrait faire appel aux mathématiques pour déterminer l'équation de la droite... Avec LINE, il suffit de donner les numéros de colonne et de ligne des deux points de l'écran que l'on veut relier.

Par exemple pour relier le coin en haut à droite (colonne 319, ligne Ø) et le coin en bas à gauche (colonne Ø, ligne 199), tapez :

```
10 CLS
20 LINE (319,0)-(0,199),3
```

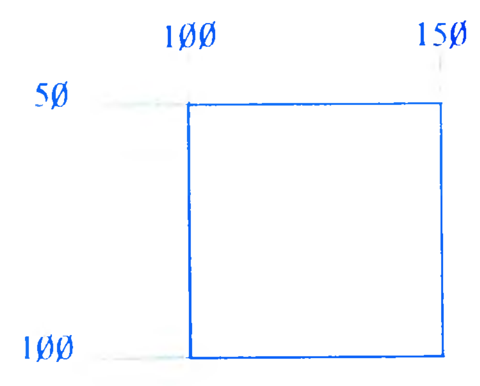
Une suite de lignes

Ajoutez maintenant la ligne suivante :

```
3Ø LINE –(319, 199), 2
RUN
```

Vous voyez un deuxième trait se dessiner à l'écran depuis le dernier point tracé (Ø, 199) jusqu'au point (319, 199). Lorsque les traits successifs sont ainsi dessinés "sans lever le crayon", il suffit dans l'instruction LINE de donner les numéros de colonne et ligne du dernier point.

Essayez de dessiner un carré rouge, par exemple entre les colonnes 1ØØ et 15Ø, et entre les lignes 5Ø et 1ØØ.

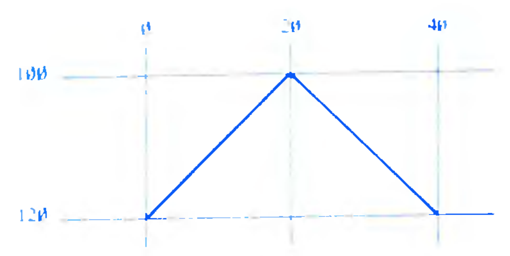


Ça y est ? Votre programme est prêt ? Vous n'avez pas oublié que vous pouvez reproduire ou modifier une ligne en changeant son numéro ?
Alors tapez-le et testez-le.

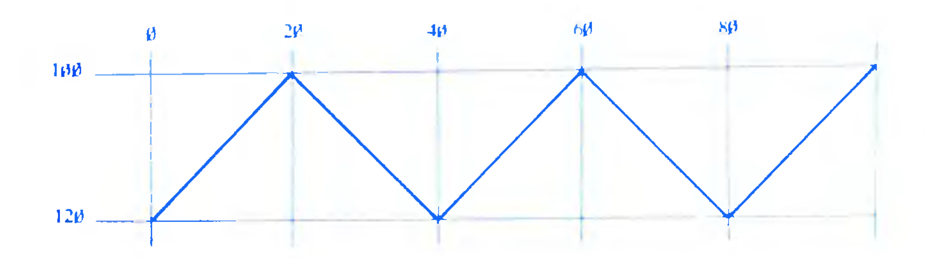
```
NEW

10 CLS
20 LINE (100,50)-(150,50),1
30 LINE -(150,100),1
40 LINE -(100,100),1
50 LINE -(100, 50),1
```

A partir d'un motif simple vous pouvez obtenir de belles figures par répétition du motif : écrivez le programme qui dessinera ces deux traits



Et maintenant comment en déduire cela ?



Répéter la même action, pour l'ordinateur cela signifie faire une boucle. Qu'est-ce qui change entre chaque boucle, c'est-à-dire entre chaque répétition du motif ?

Les numéros de lignes ne changent pas.

Le numéro de colonne de chaque point du motif augmente de 4Ø.

Nous prendrons donc comme variable COLONNE, le numéro de colonne du premier point du motif (en bas à gauche) :

```
NEW
10 CLS
20 COLONNE = 0
30 LINE (COLONNE,120)-(COLONNE+20,100)
40 LINE -(COLONNE+40,120)
50 COLONNE = COLONNE + 40
60 GOTO 30
```

La ligne 2Ø précise la position du point de départ.

Les lignes 3Ø et 4Ø dessinent le premier motif.

La ligne 5Ø donne le point de départ du deuxième motif.

La ligne 6Ø renvoie en ligne 3Ø pour dessiner le deuxième motif...

Tapez :

RUN

Lorsque le dessin est terminé à l'écran, vous entendez le léger bruit que fait l'ordinateur lorsque un programme est en train de tourner. En effet si la variable COLONNE dépasse la valeur limite 319 le spot reste toujours au même point et le programme continue de boucler. Pour l'arrêter, une seule solution : [CNT] C.

Des boîtes

De plus en plus rapide : pour dessiner un rectangle ou un carré, vous avez vu qu'il faut 4 instructions LINE, mais vous pouvez le faire encore plus vite avec une nouvelle instruction : BOX (en anglais : boîte).

Par exemple pour le même carré, il suffit de taper :

```
NEW
10 CLS
20 BOX (100,50)-(150,100),1
RUN
```

Après BOX on donne donc les numéros de deux coins opposés. Le même carré sera obtenu avec :

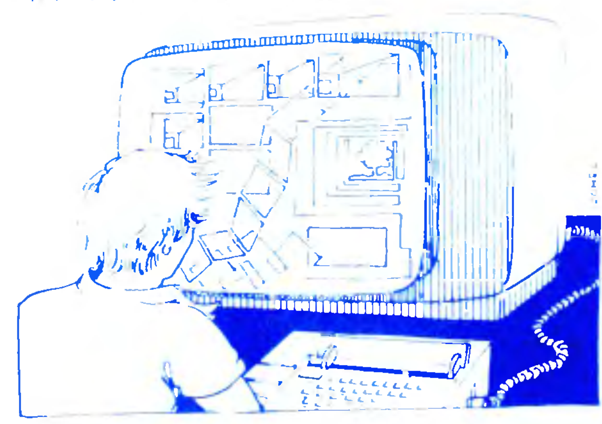
```
20 BOX (150, 50) - (100, 100), 1
```

Modifiez maintenant la ligne précédente en insérant la lettre "F" après le "X" : BOXF est une nouvelle instruction (pour "Box Full" ou boîte pleine).

```
20 BOXF (100, 50) - (150, 100), 1
RUN
```

Vous voyez un carré entièrement rouge. Il est possible de dessiner un rectangle, d'une autre couleur, à l'intérieur de celui-ci :

```
30 BOXF (110, 60) - (140, 90), 2
```



Avec les 8 couleurs dont vous disposez, vous pouvez maintenant mettre toutes les petites boîtes que vous voulez dans des grandes boîtes.

Effacer

L'instruction CLS permet d'effacer tout l'écran lorsqu'on veut afficher un nouveau dessin. Mais vous pourrez avoir envie d'effacer seulement une partie des traits, en gardant le reste. Comment faire dans ce cas ?
Vous avez remarqué dans l'exemple de la petite boîte dans la grande boîte que lorsqu'on superpose deux couleurs, c'est la dernière demandée qui l'emporte. Pour effacer un trait rouge, il suffit donc de tracer par-dessus ce trait un trait de la même couleur que le fond.

Par exemple prenons un fond noir (Ø) :

```
10 CLS
20 SCREEN 7,0,0
30 LINE (100,0)-(100,100),1
RUN
```

Et rajoutons l'instruction qui effacera le trait aussitôt qu'il est tracé :

```
4Ø LINE (1ØØ, Ø) - (1ØØ, 1ØØ), Ø
```

Encore une fois, il suffit de "doubler" la ligne précédente en changeant son numéro ainsi que le chiffre de la couleur. N'oubliez pas de vérifier le résultat de l'opération avec :

```
LIST.
RUN
```

L'exécution est très rapide : à peine le trait rouge est-il apparu à l'écran qu'il s'efface.
Cette méthode d'effacement est bonne, mais elle a un petit défaut, celui de l'encre sympathique.
En effet, si vous tapez la ligne suivante immédiatement après avoir exécuté le programme :

```
SCREEN 7, Ø, Ø
```

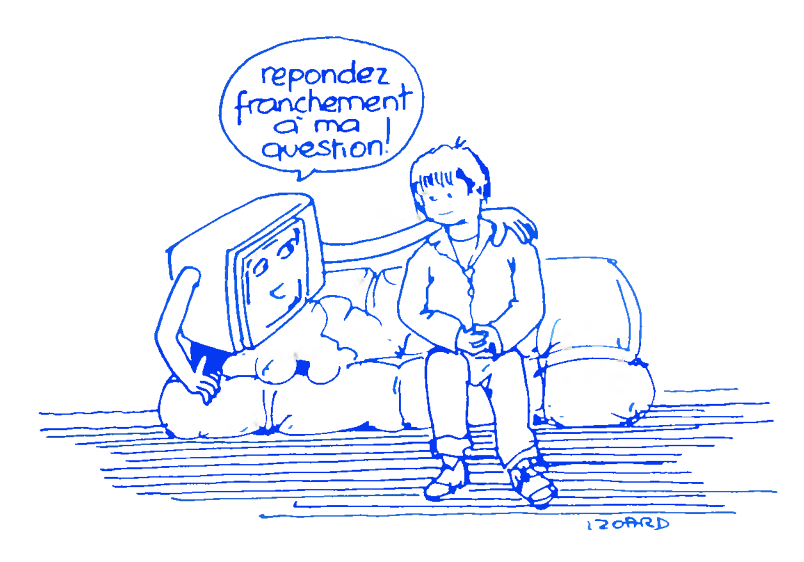
vous voyez réapparaître votre ligne verticale en blanc ! L'explication est simple. Chaque instruction PSET, LINE, BOX ou BOXF a deux effets : le premier, de marquer les points à dessiner, le deuxième de leur donner une couleur. La couleur qu'ils prennent est précisée à la fin de l'instruction, ou la couleur standard si rien n'est précisé.

Quand on trace un trait avec LINE en lui donnant la même couleur que le fond, ici Ø, les points sont marqués mais on ne les voit pas. Ils réapparaissent avec un ordre SCREEN qui a pour but de donner la même couleur à tous les points marqués de l'écran : caractères et dessins.
Pour effacer vraiment des points, il faut ôter leur marque, indiquer qu'ils retournent au fond de l'écran.
Ceci peut se faire en donnant des nombres négatifs pour les couleurs avec les conventions suivantes :
— 1 noir
— 2 rouge
— 3 vert
— 4 jaune
— 5 bleu
— 6 magenta (violet)
— 7 cyan (bleu clair)
— 8 blanc
La bonne solution pour effacer notre ligne est donc :

```
40 LINE (1ØØ, Ø)-(1ØØ, 1ØØ), -1
```

Cette fois, tout est correct. Tous les SCREEN possibles ne font rien apparaître.
De même il faut utiliser ces nombres négatifs pour effacer des points avec PSET ou des boîtes avec BOX ou BOXF.

Chapitre VII/Des programmes qui vous interrogent



Vous avez sûrement déjà vu effectuer une réservation de place de train sur un terminal d'ordinateur : l'écran se remplit de mots, de chiffres... et puis à un moment donné, l'opérateur vous demande la gare de départ, celle d'arrivée, le jour du voyage et tape ces informations au clavier.
L'exécution du programme reprend alors, l'écran affiche la réponse et éventuellement on peut encore préciser si on est amoureux du paysage et attaché au coin fenêtre ou incapable de se passer de sa cigarette.
Vous voyez tout l'intérêt d'un programme qui pose des questions pour préciser des valeurs qui peuvent changer d'une exécution à l'autre.

Voici un petit programme très simple qui vous demande votre année de naissance et calcule votre âge en 199Ø. Contrairement à Mme Irma, il ne vous dira rien d'autre.

```
NEW
10 PRINT "ANNEE DE NAISSANCE ?"
20 INPUT AN
30 PRINT "AGE EN 1990 : ";1990-AN
RUN
```

Après avoir affiché "ANNEE DE NAISSANCE ?", le curseur s'arrête sur la ligne suivante après un point d'interrogation. Ce point d'interrogation indique que l'ordinateur attend votre réponse : le programme s'est arrêté en ligne 2Ø.

Allez-y : tapez votre année de naissance et n'oubliez pas de taper ensuite sur ENTREE pour que l'ordinateur enregistre cette valeur. L'exécution reprend alors et la réponse s'affiche sur la ligne suivante.

C'est l'instruction INPUT ("introduisez") qui affiche le point d'interrogation à l'écran et donne à la variable AN le nombre que vous avez tapé au clavier. Refaites RUN : à nouveau un point d'interrogation indique que l'ordinateur attend une réponse. Vous pouvez donner une nouvelle année de naissance.

Si vous tapez une lettre, l'ordinateur n'accepte pas la réponse et affiche :

?Redo ("recommencez").

Il attend un nombre et rien d'autre. Tant que vous ne fournirez pas un nombre comme réponse, vous obtiendrez ce message d'erreur. En particulier, vous ne pouvez pas demander LIST ou RUN, car le programme est en cours d'exécution.

Si vous ne voulez vraiment pas répondre à la question, il faut arrêter le programme avec CNT C.

Ne trouvez-vous pas que cela serait plus joli si le nombre tapé au clavier s'affichait à la suite de la question ? Ajoutez donc un point-virgule à la fin de la ligne 1Ø :

```
1Ø PRINT "ANNEE DE NAISSANCE ?";
RUN
```

Mais alors, il y a 2 points d'interrogation. Tapez un nombre quelconque pour finir le programme puis supprimez le point d'interrogation de la ligne 1Ø :

```
1Ø PRINT "ANNEE DE NAISSANCE";
```

Le BASIC du TO7 accepte un raccourci de langage dans l'utilisation de l'instruction INPUT. Vous pouvez placer le message "ANNEE DE NAISSANCE" immédiatement après INPUT, en le séparant du nom de la variable par un point-virgule :

```
1Ø INPUT "ANNEE DE NAISSANCE";AN
```

Effacez la ligne 2Ø en tapant :

2Ø (suivi de ENTREE).

Vous avez gagné une ligne.

Si le point d'interrogation ne vous convient pas, il est possible de ne pas l'afficher en mettant une virgule au lieu d'un point-virgule entre le message et la variable :

```
1Ø INPUT "ANNEE DE NAISSANCE",AN
```

Choisir sa couleur

L'ordinateur peut vous demander de répondre par un nombre sans que ce nombre soit destiné à faire des calculs. Par exemple Ø et 1 peuvent distinguer les fumeurs des non-fumeurs; les nombres de Ø à 7 peuvent représenter les couleurs disponibles sur votre micro-ordinateur.

Entrez le programme suivant :

```
NEW
10 CLS
20 INPUT "QUELLE COULEUR ";COULEUR
30 BOXF (0,100)-(110,200),COULEUR
RUN
```

La ligne 2Ø affecte à la variable COULEUR le nombre que vous tapez au clavier et la ligne 3Ø affiche un rectangle dans la couleur choisie.
Si vous répondez 6, le rectangle sera bleu clair.
Si vous répondez 9, vous obtenez un message d'erreur :

```
?FC Error in 3Ø
```

C'est parce que vous avez utilisé un chiffre ne correspondant à aucune couleur dans l'instruction BOXF.

Videz l'écran avec la touche [RAZ] avant de faire tourner à nouveau le programme :

```
RUN
```

La question vous est à nouveau posée. Si vous voulez que la question vous soit reposée systématiquement après chaque essai, rajoutez une ligne :

```
4Ø GOTO 1Ø
```

Vous avez à peine le temps de voir le rectangle qu'il est déjà effacé. Nous verrons plus loin (chap. XI) comment le faire durer.

Le programme boucle. Lorsque vous en avez assez, arrêtez avec [CNT] C.

Créer son drapeau

De nombreux drapeaux sont formés de trois bandes verticales de couleurs différentes. Le programme suivant vous permettra de comparer les différentes possibilités avec les 8 couleurs du TO7 :

```
1Ø CLS
2Ø INPUT "COULEUR BANDE 1 ";A
3Ø INPUT "COULEUR BANDE 2 ";B
4Ø INPUT "COULEUR BANDE 3 ";C
5Ø BOXF (Ø,Ø)-(11Ø,2ØØ),A
6Ø BOXF (11Ø,Ø)-(21Ø,2ØØ),B
7Ø BOXF (21Ø,Ø)-(32Ø,2ØØ),C
```

Remarquez que les numéros de colonnes et de lignes maximaux ont été pris égaux à 32Ø et 2ØØ (au lieu de 319 et 199) par simplification d'écriture.

RUN

Vous pouvez maintenant créer votre drapeau.

Ce programme peut être raccourci en regroupant les 3 instructions INPUT en une seule : effacez les lignes 3Ø et 4Ø et modifiez ainsi la ligne 2Ø :

```
2Ø INPUT "COULEURS DESIREES"; A, B, C
```

Faites :

```
LIST
```

pour vérifier la modification, puis :

```
RUN
```

Lorsque le point d'interrogation apparaît à l'écran, vous devez donner les 3 valeurs des couleurs choisies en les séparant par des virgules. Le premier chiffre entré sera affecté à la variable A, le deuxième à B et le troisième à C.

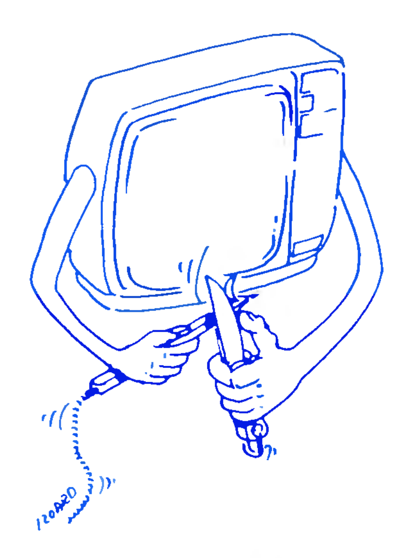
N'oubliez surtout pas de mettre des virgules, sinon vous aurez droit au message :

```
?Redo
```

et il faudra répondre à nouveau en donnant les trois couleurs. C'est la même chose si vous ne donnez que deux chiffres sur trois : même message d'erreur et il faut répondre à nouveau.

Les ordinateurs sont patients mais ce sont des êtres bien pointilleux ... sur les virgules. Un conseil, ne mettez pas trop d'INPUT multiples dans vos programmes ou si vous le faites, n'oubliez pas de bien préciser dans la question combien de réponses sont attendues.

Chapitre VIII/Le crayon optique



Le crayon optique ne vous a servi jusqu'à présent qu'à choisir en début de séance entre le BASIC et le réglage de ce crayon optique. C'est un peu mince. Vous allez pouvoir maintenant l'utiliser comme instrument de dialogue avec l'ordinateur et comme instrument de dessin à l'écran.

Si vous n'avez pas réglé votre crayon, appuyez sur le bouton d'initialisation pour faire apparaître le "menu" à l'écran. Réglez alors votre crayon en appuyant la pointe au centre du carré et demandez à nouveau le BASIC.

Lorsque vous appuyez en un point de l'écran avec le crayon optique l'ordinateur sait quels sont les numéros de colonne et de ligne de ce point.
Si vous aussi vous voulez le savoir, c'est l'instruction INPUTPEN qui vous le dira. Tapez le petit programme suivant où C et L désignent les variables contenant les numéros de colonne et de ligne du point désigné à l'écran :

```
10 CLS
20 INPUTPEN C,L
30 PRINT "COLONNE";C;"LIGNE";L
RUN
```

Rien ne se passe.
En effet en ligne 20, l'ordinateur attend que le crayon optique désigne un point à l'écran. L'exécution reste interrompue tant que cela n'est pas fait.

Vous avez déjà rencontré le même genre de situation avec l'instruction INPUT : l'ordinateur attend indéfiniment une réponse de l'utilisateur, mais dans ce cas par l'intermédiaire du clavier.
Appuyez donc en un point de l'écran avec le crayon optique, vous voyez alors s'afficher les coordonnées du point : par exemple

COLONNE 152 LIGNE 38

Peut-être avez-vous obtenu une réponse déconcertante :

COLONNE −1 LIGNE −1

Que signifient ces nombres −1 alors qu'un numéro de colonne est compris entre Ø et 319 ?
Ils veulent simplement dire que l'ordinateur n'a pas pu donner de numéros au point choisi :
— ou bien parce qu'il est en-dehors du rectangle qui forme la partie utile de l'écran,
— ou bien parce que votre écran n'envoie pas assez de lumière sur le crayon optique. Il faut dans ce cas augmenter la luminosité du téléviseur.
A chaque fois que vous utilisez le crayon optique prenez l'habitude de pousser un peu la luminosité et évitez les fonds noir ou rouge.
Pour connaître les coordonnées d'un nouveau point, il vous faut faire à nouveau :

RUN

Voyez-vous comment l'éviter ?
Il serait plus agréable qu'après avoir imprimé les coordonnées d'un point, le programme soit prêt à enregistrer un nouveau point.
Çà y est, vous y êtes ?
Il suffit de faire boucler le programme en ajoutant la ligne suivante :

```
4Ø GOTO 2Ø
RUN
```

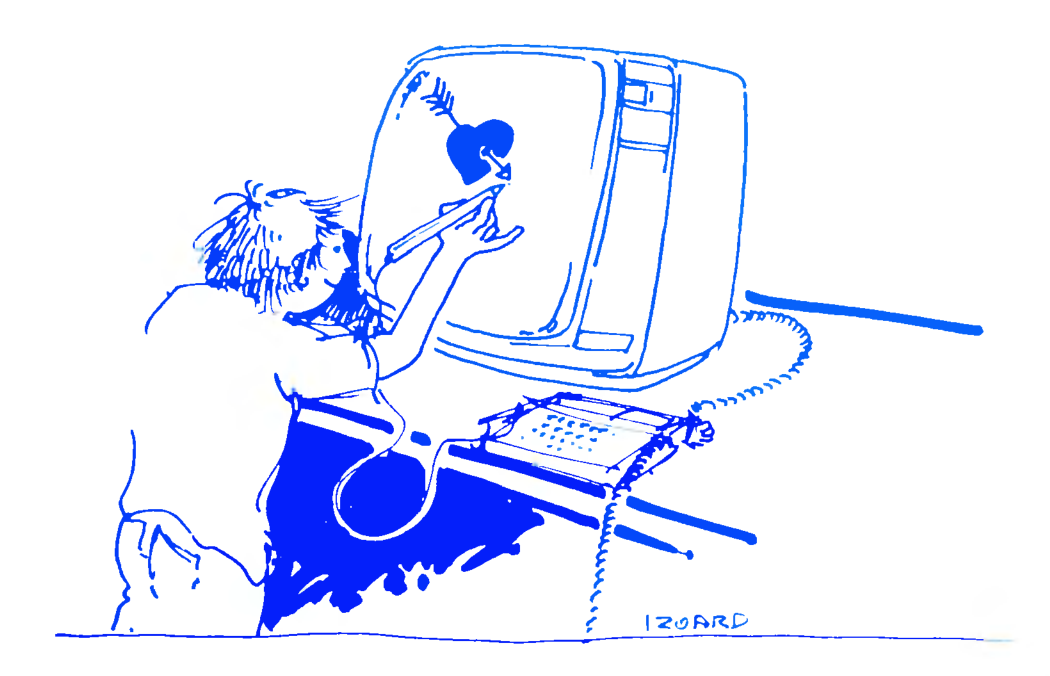
Vous voyez alors défiler rapidement les numéros des points successifs montrés à l'écran. Pour vider l'écran, la touche [RAZ] reste sans effet : il faut d'abord arrêter le programme avec [CNT] C.
Essayez de suivre avec le crayon optique une colonne à partir du haut : le numéro de colonne restera le même (ou du moins variera peu) tandis que les numéros de lignes vont croître de Ø à 199.

Essayez de même de suivre une ligne horizontale à partir de la gauche. Cette fois le numéro de ligne reste le même tandis que les numéros de colonnes augmentent. Sur TO7 uniquement, vous constatez que les numéros de colonnes ne prennent pas toutes les valeurs, ils augmentent de 8 en 8 à partir de 3 :
3, 11, 19, 27, 35 ....

Les colonnes Ø, 1, 2, 3, 4, 5, 6, 7 reçoivent toutes le même numéro : 3
Les colonnes 8, 9, 1Ø, 11, 12, 13, 14, 15 reçoivent toutes le même numéro : 11
Les colonnes 16, 17, 18, 19, 2Ø, 21, 22, 23 reçoivent toutes le même numéro : 19
et ainsi de suite ...

Comment écrire sur sa télévision ?

Vous avez l'habitude de regarder votre écran de télévision de façon plus ou moins passive sauf si, étant d'un naturel violent, vous lancez sur votre poste ce qui vous tombe sous la main lorsque les programmes TV ne vous satisfont pas.
Maintenant vous pourrez être actif et remplir vous-même l'écran de vos propres œuvres.



Vous souvenez-vous de l'instruction qui permet d'afficher un point à l'écran ?

Le point à l'intersection de la colonne C et de la ligne L s'allumera avec :

PSET (C, L)

Pour que ce point s'allume lorsque l'on appuie dessus avec le crayon optique, on précise les valeurs des variables C et L avec l'instruction INPUTPEN :

```
10 CLS
20 INPUTPEN C,L
30 PSET (C,L)
```

Remarquez que les coordonnées du point doivent être entre parenthèses après PSET et sans parenthèses après INPUTPEN. Nous vous avons déjà dit que l'ordinateur aime bien que l'on soit précis.

Vous pouvez bien sûr changer la couleur du point, par exemple en vert :

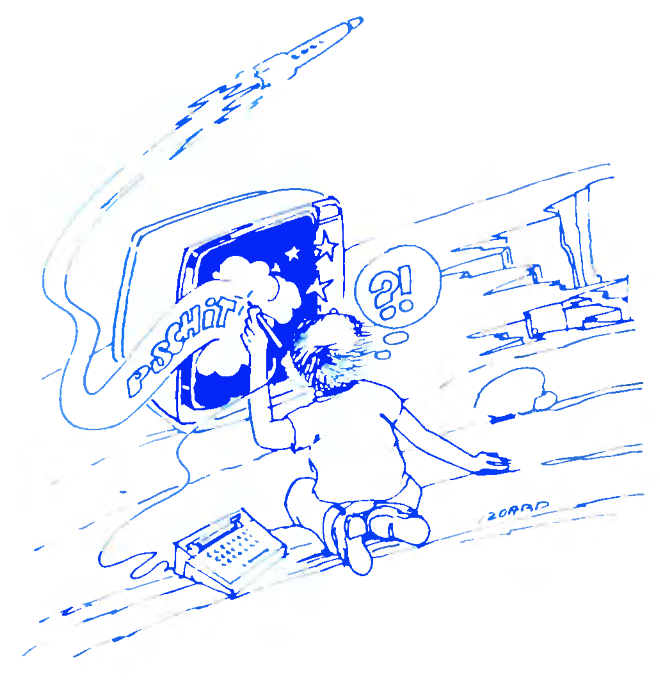
30 PSET (C, L), 2

Un seul point, c'est un peu mince. Comment en dessiner autant que vous voulez ?
Evidemment il est toujours possible de refaire :

RUN

après chaque point.
Mais maintenant vous avez l'habitude. Pour dire à l'ordinateur "Recommence", il suffit de faire boucler le programme :

40 GOTO 20
RUN



Vous pouvez alors représenter une nuit constellée d'étoiles.
Sur TO7 uniquement, si vous maintenez la pression sur le crayon optique en le déplaçant sur l'écran, vous voyez se dessiner un trait en pointillé. Ceci provient du fait que l'instruction INPUTPEN ne définit les numéros de colonnes que de 8 en 8.
Si vous tracez un trait horizontal, vous voyez les points qui sont situés sur les colonnes 3, 11, 19, 27... Par contre si vous tracez un trait vertical, vous obtenez une ligne continue sur la plus proche de ces colonnes 3, 11, 19...

Mais rassurez-vous ceci n'implique pas que vos dessins seront toujours en pointillés. C'est seulement une possibilité que vous pourrez utiliser.

Dessiner à jet continu

Pour dessiner en continu à l'écran, il faudrait joindre directement deux points à l'écran. C'est donc l'instruction LINE que nous allons utiliser. Rappelons son fonctionnement :

LINE (A, B) – (C, D)
joint les points (A, B) et (C, D)
LINE – (C, D)
joint le dernier point affiché à l'écran au point (C, D)

Vous pouvez maintenant essayer d'écrire le programme qui joint à l'écran les points qui sont appuyés avec le crayon optique. Rédigez votre programme sur une feuille de papier et faites-le tourner dans votre tête avant de le faire tourner dans l'ordinateur.

Testez ensuite ce programme. Si le résultat obtenu à l'écran est celui que vous espériez, bravo !

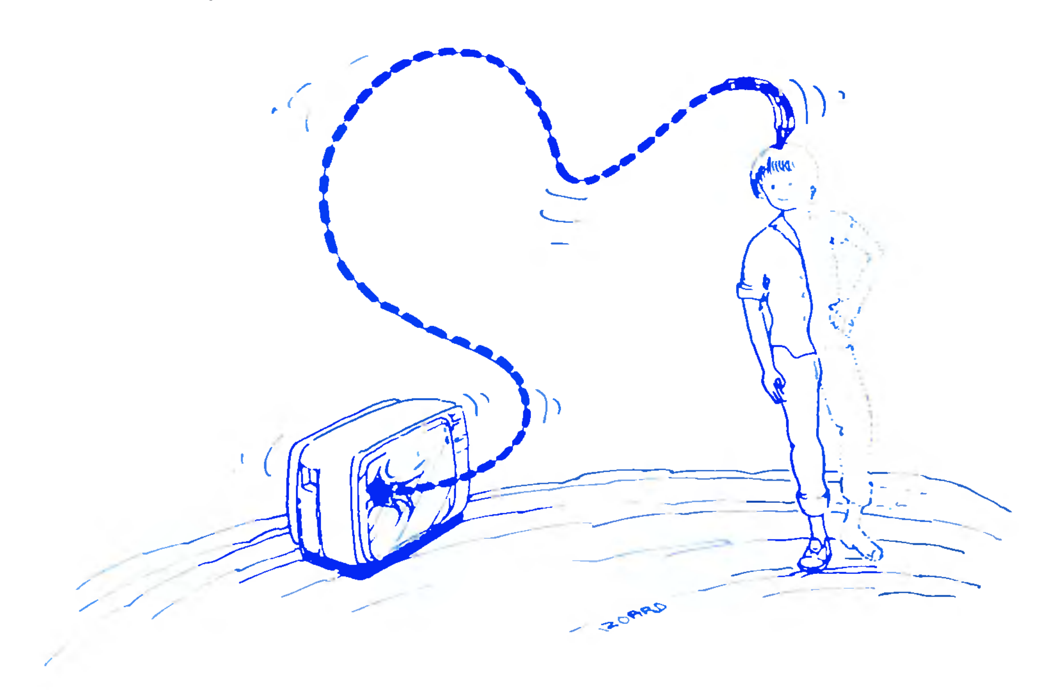
Sinon voici quelques indications :
— il faut d'abord allumer le premier point appuyé à l'écran
— ensuite introduire les coordonnées du deuxième point
— joindre ces deux points
— introduire les coordonnées d'un nouveau point et le joindre au précédent

Lorsque les programmes se compliquent, cela gagne toujours du temps de définir ainsi par écrit et en français ce que doit faire le programme et dans quel ordre.

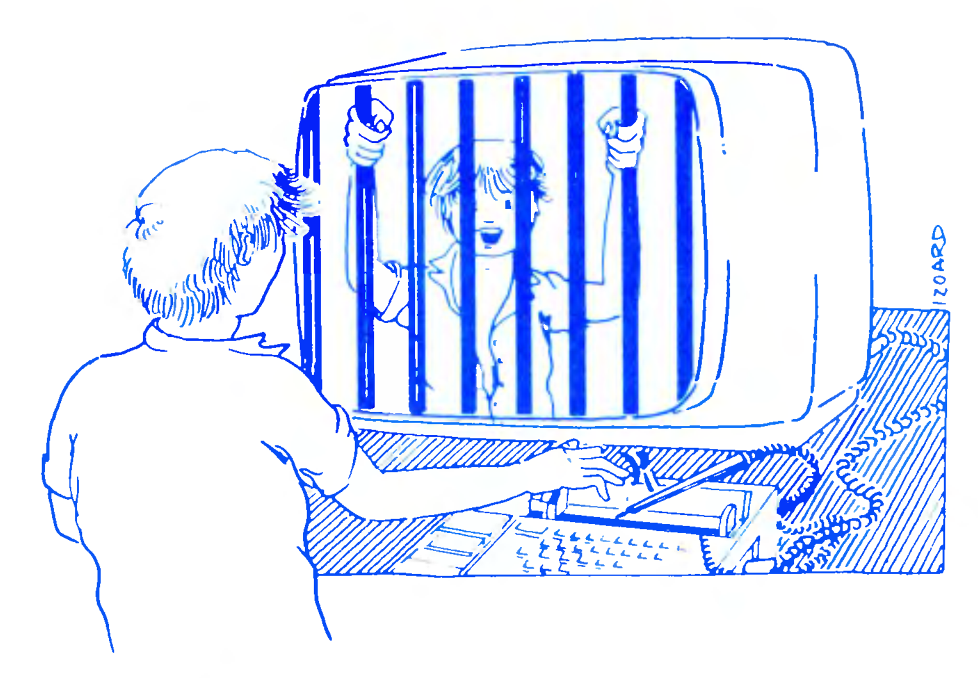
Voici la traduction en BASIC :

```
10 CLS
20 INPUTPEN A,B
30 PSET(A,B)          allumer le premier point
40 INPUTPEN C,D       introduire les coordonnées du 2e point
50 LINE -(C,D)        le joindre au précédent
60 GOTO40             recommencer avec un nouveau point
```

Vous pouvez maintenant dessiner en continu, c'est-à-dire sans lever le crayon.



Chapitre IX/Avec des si....



Arrêter une boucle quand on le désire

Essayons de rédiger un programme qui dessine une série de traits verticaux, c'est-à-dire une grille. Chaque trait vertical est caractérisé par son numéro de colonne. Ce numéro est donc une variable, appelons-la COL.
La colonne COL sera dessinée avec l'instruction :

LINE (COL, Ø) – (COL, 199)

Si on veut espacer les barreaux de 1Ø en 1Ø, pour chaque nouvelle colonne, la variable COL sera remplacée par COL + 1Ø
Il suffit maintenant de préciser le numéro de la première colonne de la grille, par exemple 1ØØ.
Tous les éléments du programme sont en place. Vous pouvez donc le rédiger.

```
10 CLS
20 COL = 100
30    LINE (COL,0)-(COL,199),1
40    COL = COL + 10
50    GOTO 30
```

Lorsque l'écran est rempli de barreaux parallèles à partir de la colonne 1ØØ, le programme boucle encore même s'il ne se passe plus rien à l'écran.

En effet dès que la variable COL dépasse la valeur 319, la ligne 3Ø du programme entraîne l'affichage de la dernière colonne 319.
Une fois qu'elle est affichée, il ne se passe donc plus rien. Pour arrêter il faut faire CNT C.

Vous pouvez modifier la distance entre deux barreaux en changeant la valeur 1Ø dans la ligne 4Ø.
Par exemple prenez :

```
4Ø COL = COL + 1
```

l'écran se remplit petit à petit à partir de la colonne 1ØØ.

Il est possible d'arrêter la progression de cette bande à l'écran, c'est-à-dire d'arrêter le programme, avec CNT C. Mais si l'on veut remplir l'écran entre les colonnes 1ØØ et 199, il sera impossible de s'arrêter exactement sur la colonne 2ØØ.

Pour faire cela nous allons utiliser une nouvelle instruction :
IF ... THEN (si.... alors).
Grâce à cette instruction nous allons pouvoir dire à l'ordinateur en fin de chaque boucle (donc en ligne 5Ø) :
"va tracer la nouvelle colonne seulement si son numéro est inférieur à 2ØØ".

Le symbole pour "inférieur à" est "<", il se trouve sur la dernière ligne du clavier.
La fin de la boucle s'écrira donc :

```
5Ø IF COL<2ØØ THEN GOTO 3Ø
```

ou plus simplement encore :

```
5Ø IF COL<2ØØ THEN 3Ø
```

car le GOTO n'est pas obligatoire.
Cette ligne 5Ø se lit :
"SI la variable COL est inférieure à 2ØØ ALORS aller à la ligne 3Ø".

Ainsi à chaque tour, au passage par la ligne 5Ø, l'ordinateur teste si la nouvelle valeur de COL est encore inférieure à 2ØØ. Lorsque COL atteint la valeur 199, la colonne correspondante est tracée, mais au tour suivant COL devient égal à 2ØØ.
La condition COL<2ØØ n'est plus réalisée et l'ordinateur saute la conséquence "GOTO 3Ø" pour aller à la ligne suivante. Et comme il n'y a plus de ligne, il s'arrête.

Le programme boucle donc tant que la condition COL<2ØØ est réalisée. Dès que cette condition n'est plus réalisée, il saute à la

ligne suivante. La forme générale de l'instruction IF...THEN... est :
IF une certaine condition THEN numéro de ligne.

Comparer deux nombres

Dans l'exemple précédent, lorsque la condition était réalisée, il fallait effectuer toute une série d'instructions. Cela obligeait à dire à l'ordinateur : "Remonte exécuter la séquence précédente".

Mais parfois l'opération consiste en une seule instruction simple. Il est alors inutile de faire une rupture dans le déroulement du programme : on peut écrire l'instruction à effectuer directement après THEN.

Regardez le programme suivant :

```
NEW
10 INPUT "DONNEZ 2 NOMBRES (POSITIFS OU
NEGATIFS)"; A,B
20 IF A<B THEN PRINT A;"<";B
30 PRINT "***"
```

La ligne 1Ø entre les données, c'est-à-dire les nombres A et B.
La ligne 2Ø effectue le test :
— si la condition A < B est réalisée, alors le programme affiche le nombre A puis le caractère "<", puis enfin le nombre B.
Ensuite le programme continue à la ligne 3Ø.
— si la condition A < B n'est pas réalisée, le programme saute directement à la ligne 3Ø.
Ainsi dans les deux cas, le programme suit les lignes dans l'ordre croissant.

Cette nouvelle manière d'utiliser l'instruction IF...THEN peut donc se schématiser ainsi :
IF condition THEN instruction

Et alors sinon ?

Le programme précédent vous a sans doute laissé sur votre faim. En effet si A est plus grand que B, l'ordinateur n'affiche rien. Pour remédier à cela il faut lui dire :
SI A < B ALORS imprimer "A < B" SINON imprimer "A > B"

En BASIC cela se dit :

IF A<B THEN PRINT A;“<”;B | ELSE PRINT A;“>”;B |

Rajoutez la partie encadrée à la fin de la ligne 2Ø puis exécutez le programme :

```
1Ø INPUT "DONNEZ 2 NOMBRES (POSITIFS OU
NEGATIFS)"; A,B
2Ø IF A<B THEN PRINT A;"<";B ELSE PRINT
A;">";B
3Ø PRINT "***"
RUN
```

Avec A = 3 et B = 5 vous obtenez 3<5
Avec A = −9 et B = −11 vous obtenez −9>−11

N'avons nous pas oublié un cas particulier ?
Si vous voulez prendre l'ordinateur en défaut, demandez-lui de comparer deux nombres égaux, par exemple 5 et 5. Pouvez-vous prévoir la réponse avant de faire tourner le programme ?

En ligne 2Ø, l'ordinateur effectue le test A<B. Comme cette condition n'est pas réalisée, alors il exécute l'instruction qui suit ELSE c'est-à-dire PRINT A>B, puis il poursuit à la ligne suivante. Le programme affichera donc :

```
5>5
***
```

L'instruction IF...THEN...ELSE marche donc seulement lorsqu'il y a deux cas possibles. Sa forme générale est :
IF condition THEN première instruction ELSE deuxième instruction

Des si les uns à la suite des autres

Comment faire lorsqu'il y a plusieurs tests à réaliser ?
Dans la comparaison de deux nombres, il y a trois cas possibles :
A<B
A>B
A=B
Il suffit alors d'effectuer les tests les uns à la suite des autres :

```
2Ø IF A<B THEN PRINT A;“<”;B
22 IF A>B THEN PRINT A;“>”;B
24 IF A=B THEN PRINT A;“=”;B
```

Dans chaque cas, lorsque la condition n'est pas réalisée le programme saute à la ligne suivante et regarde la condition suivante.
Pour chaque nouveau couple de nombres, le programme effectue donc les trois tests. Ceci n'est peut-être pas la méthode la plus rapide mais elle présente l'avantage de donner une rédaction très claire du programme.

Chapitre X/Un peu de logique



Tester une case à l'écran

Lorsque vous allumez l'ordinateur, la machine vous propose de choisir entre le BASIC et le réglage du crayon optique en appuyant sur des cases à l'écran.
Il est souvent utile de demander ainsi à celui qui utilise l'ordinateur de choisir entre plusieurs possibilités avec le crayon optique.

Ainsi peut-on demander :
☐ Voulez-vous encore jouer à ce jeu ?
☐ Voulez-vous un autre jeu ?
☐ Voulez-vous arrêter ?

Essayons de rédiger un programme qui, pour commencer, travaille avec une seule case : si l'on appuie le crayon optique dans la case, l'écran s'efface et on entend "BIP".
Vous pourrez l'utiliser ultérieurement dans des programmes plus importants. Tout "gros" programme comporte souvent une série de petits programmes indépendants qui peuvent être ainsi utilisés à diverses occasions.

Quelles seront les différentes étapes du programme ?
— afficher au milieu de l'écran une phrase du genre "Pour entendre BIP, appuyer au centre de la case"
— dessiner la case
— entrer les coordonnées du point que l'utilisateur désigne à l'écran avec le crayon optique

— voir si ce point est à l'intérieur ou à l'extérieur de la case en comparant ses coordonnées aux numéros de ligne et de colonne des bords de la case

- si le point est à l'extérieur, ne rien faire et attendre qu'un nouveau point soit désigné pour entrer ses coordonnées
- si le point est à l'intérieur, faire "BIP" et effacer l'écran

Précisons les deux instructions qui vous manquent pour écrire ce programme :
— la première pour faire "BIP" est tout simplement "BEEP", anglais oblige !
Vous pouvez la tester en mode direct en tapant BEEP. Si vous voulez jouer à la mitraillette, essayez :

```
100 BEEP
110 GOTO 100
```

— la deuxième pour imprimer un texte à partir d'une position quelconque de l'écran est LOCATE.
Tapez le programme suivant pour en voir le rôle :

```
NEW
100 LOCATE 10,10
110 PRINT "ICI"
```

Puis remplacez les deux "1Ø" qui suivent l'instruction LOCATE par d'autres valeurs :

```
1ØØ LOCATE  5, 1Ø
RUN
1ØØ LOCATE 2Ø, 1Ø
RUN
1ØØ LOCATE 1Ø, 5
RUN
1ØØ LOCATE 1Ø, 2Ø
```

Cette instruction fixe donc la position du premier caractère de la phrase à imprimer soit "I".
Ce caractère, comme tous les autres, est dessiné à l'intérieur d'une case 8×8. Il y a donc 32Ø/8 = 4Ø caractères dans une ligne et 2ØØ/8 = 25 lignes de caractères à l'écran.
Dans l'instruction LOCATE, le deuxième chiffre fixe la ligne (comprise entre Ø et 24) et le premier chiffre fixe la position du caractère sur la ligne (comprise entre Ø et 4Ø).

Vous pouvez maintenant prendre une feuille de papier pour écrire le programme.

Voici notre solution.
Nous avons placé la case au milieu de l'écran, ses dimensions étant 8×8.
En outre cette case est placée exactement à la place d'un caractère : sa ligne de départ (96) et sa colonne de départ (16Ø) sont des multiples de 8.

Pour l'esthétique, nous avons entouré la case d'un pourtour de couleur différente comme dans le réglage du crayon optique :

```
10 CLS
20 LOCATE 10,8
30 PRINT "POUR ENTENDRE UN BIP"
35 LOCATE 6,10
40 PRINT "APPUYER AU CENTRE DE LA CASE"
50 BOX (159,95)-(168,104),1
60 BOXF(160,96)-(167,103),2
70 INPUTPEN C,L
80 IF C>167 THEN 70
90 IF C<160 THEN 70
100 IF L>103 THEN 70
110 IF L<96  THEN 70
120 BEEP
130 CLS
RUN
```

Pour que le programme arrive en ligne 12Ø il faut qu'aucune des quatre conditions : C > 167, C < 16Ø, L > 1Ø3, L < 96, ne soit réalisée. Si une seule de ces conditions est réalisée, le point est en dehors de la case et il faut aller attendre un nouveau point en ligne 7Ø.
On peut regrouper les 4 conditions sur une seule ligne en les reliant par des "ou" :

```
8Ø IF C>167 OR C<16Ø OR L>1Ø3 OR L<96 THEN 7Ø
```

Vous voyez en tapant cette ligne qu'elle s'imprime sur 2 lignes à l'écran. Ceci ne pose pas de problèmes comme nous l'avons déjà vu : une ligne de programme peut contenir jusqu'à 255 caractères alors qu'une ligne à l'écran n'en comporte que 4Ø.
Cependant pour faciliter la lecture du programme, nous décomposerons cette ligne en deux :

```
8Ø IF C>167 OR C<16Ø THEN 7Ø
9Ø IF L>1Ø3 OR L<96 THEN 7Ø
```

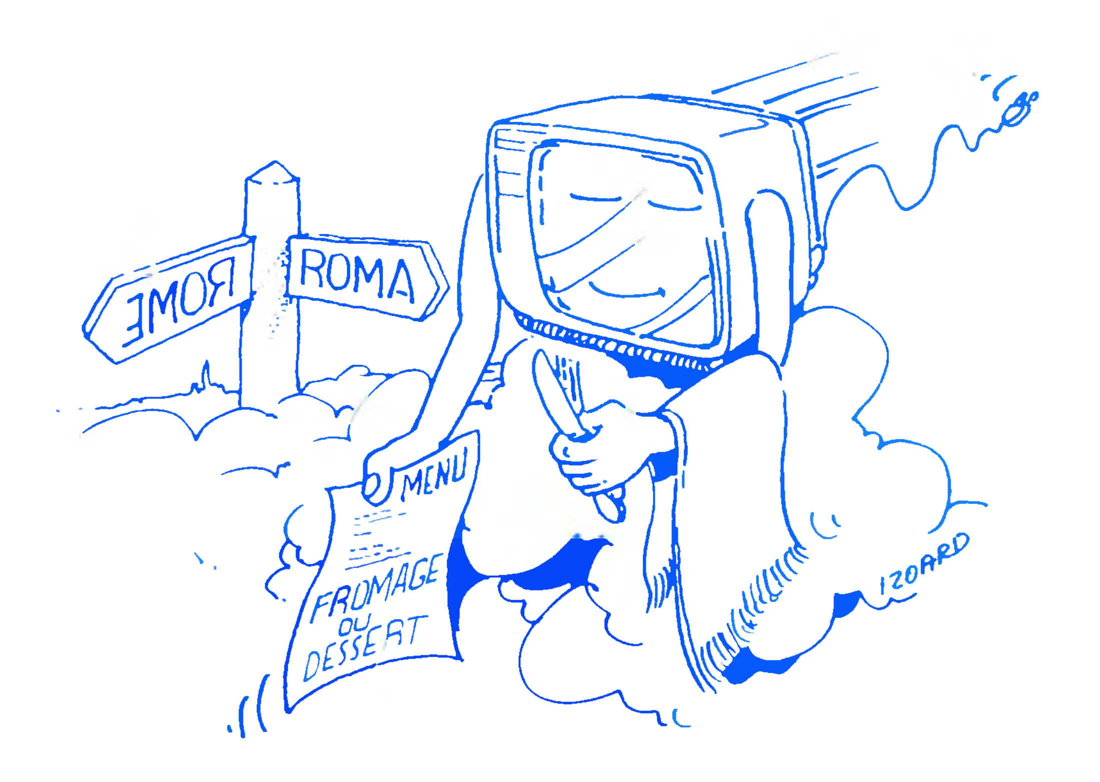
effacez les lignes 1ØØ et 11Ø :

```
1ØØ
```

```
11Ø
RUN
```

Ce programme est bien équivalent au précédent.

Comment remplacer des "ou" par des "et"



Revenons au programme précédent. Tous les chemins mènent à Rome et peut-être avez-vous pensé à rédiger le test sous la forme suivante :

"SI C et L sont compris entre les limites de la case, ALORS effectuer les instructions BEEP et CLS."

Pour que le point soit à l'intérieur de la case, il faut que les 4 conditions suivantes soient réalisées à la fois :

C ⩾ 16Ø et C ⩽ 167 et L ⩾ 96 et L ⩽ 1Ø3

Les symboles ⩾ et ⩽ utilisés en mathématiques signifient "inférieur ou égal à" et "supérieur ou égal à". Ils s'écrivent en BASIC : < = et > =, c'est-à-dire en frappant les deux signes successivement.

Le programme peut donc encore s'écrire :

```
10 CLS
20 LOCATE 10,8
30 PRINT "POUR ENTENDRE UN BIP"
35 LOCATE 6,10
40 PRINT "APPUYER AU CENTRE DE LA CASE"
50 BOX (159,95)-(168,104),1
60 BOXF(160,96)-(167,103),2
70 INPUTPEN C,L
80 IF C >= 160 AND C <= 167 AND L >= 96
AND L <= 103 THEN 100
90 GOTO 70
100 BEEP
110 CLS
```

Il nous a fallu rajouter une instruction supplémentaire : si les 4 conditions sont réalisées alors on va en ligne 1ØØ, sinon le programme continue en ligne 9Ø d'où il est renvoyé en ligne 7Ø pour attendre un nouveau point.

Tout ce qu'il faut pour faire des comparaisons

Lorsque vous aurez deux nombres à comparer, voici les opérateurs que vous pourrez utiliser :

=	égal à
< >	différent de
>	supérieur à (et pas égal)
<	inférieur à (et pas égal)
> =	supérieur ou égal à
< =	inférieur ou égal à

Et vous pourrez regrouper les conditions avec les deux conjonctions :

OR	ou
AND	et

Ceci permet d'exprimer aussi bien une condition que son opposée. C'est ce que nous avons utilisé dans les deux versions du programme précédent.

A = 5	a pour opposé	A< >5
B> =6	”	B<6
A>1 AND A<2	”	A< =1 OR A> =2

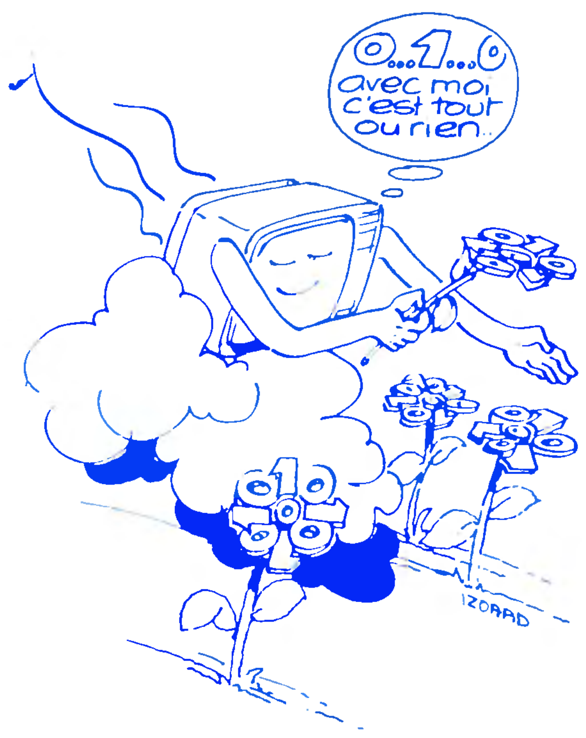
Essayez d'écrire les expressions opposées aux expressions suivantes :

A<2 AND A>Ø
D< >7
B> =4 OR B< =6

Vous avez aussi la possibilité d'écrire les conditions sous une forme négative en utilisant NOT.

NOT (A = 5)	est équivalent à	A< >5
NOT (B> =6)	,,	B<6
NOT (A>1 AND A<2)	,,	A < =1 OR A> =2

Chapitre XI/La répétition contrôlée



De 2 en 2

L'ordinateur est une machine binaire, c'est bien connu. Il raisonne en tout ou rien, ou presque. Il n'est pas question de s'engager ici dans les joies du calcul binaire, où l'on n'emploie que des Ø et des 1 pour faire les opérations.

Cependant même du côté de l'usager qui dialogue en BASIC avec sa machine on peut voir des traces de l'importance de ce mode de calcul. Ainsi les mémoires des micro-ordinateurs sont-elles de 1K, 4K, 8K, 16K, 32K, 64K... c'est-à-dire des multiples de 2. On pourrait penser que 1Koctets représente 1000 octets, de même que 1km représente 1000 mètres. En fait 1K en informatique signifie 1Ø24.

Ce nombre 1Ø24 est égal à 2 à la puissance 1Ø :
1Ø24 = 2*2*2*2*2*2*2*2*2*2 ou 1Ø24 = 2^1Ø

Les puissances de 2 se retrouvent partout en informatique : l'octet lui-même au niveau le plus élémentaire de la mémoire est formé de 8 "bits", or 8 est encore égal à une puissance de 2 :
8 = 2*2*2 ou 8 = 2^3

Comment écrire un programme qui calcule les puissances de 2, par exemple jusqu'à 2 à la puissance 1Ø ?

Évidemment il faut trouver mieux que le programme formé par la suite de 1Ø instructions :

```
PRINT 2^1
PRINT 2^2
PRINT 2^3
.......
```

Votre programme est prêt sur le papier ? Alors testez-le.
Non ?
Pour chaque puissance de 2, on refait la même opération en changeant simplement la valeur de la puissance. Cette phrase "changer simplement la valeur de ...." a dû faire TILT dans votre esprit et provoquer le réflexe : "faisons de cette valeur une variable appelée .....".

Appelons donc P la puissance :
— au premier tour P vaut 1 et à chaque tour suivant P augmente d'une unité.
— il faut alors tester si P est encore inférieur ou égal à 1Ø :
- si c'est le cas, il faut recommencer un tour avec P + 1
- sinon le programme est terminé

```
20 P = 1
30    PRINT 2^P
40    P = P + 1
50    IF P<=10 THEN 30
RUN
```

La suite des puissances de 2 s'affiche à l'écran. Le résultat sera plus facile à lire si chaque nombre est précédé de sa signification :

```
3Ø PRINT "2 PUISSANCE";P;"EGALE:";2^P
```

Cette ligne affiche à la suite les caractères entre guillemets et les nombres séparés par des points-virgules.

Commenter le programme

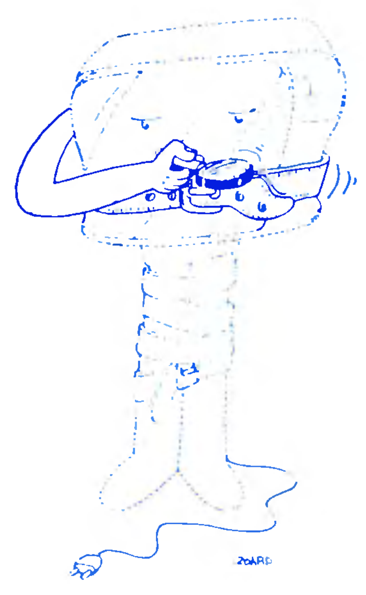
Ajoutons maintenant la ligne suivante au début du programme :

```
1Ø REM PUISSANCES DE 2
```

L'instruction REM (pour REMARK) permet de mettre des commentaires dans votre programme. En effet, tout ce qui est écrit après le REM jusqu'à la fin de la ligne sera ignoré à l'exécution.

Nous allons prendre l'habitude de donner un titre à nos programmes avec cette instruction. Ceci nous permettra de mieux les retrouver plus tard quand ils seront plus nombreux.
Cette instruction REM peut être remplacée par une simple apostrophe ' qui joue le même rôle :
1Ø ' PUISSANCES DE 2

Le contrôle de la boucle



Lorsque l'on sait à l'avance combien de fois le programme doit effectuer une boucle, par exemple ici 1Ø fois, on utilise une nouvelle instruction qui évite d'écrire le test IF ... THEN.
Ainsi le programme précédent s'écrit de manière équivalente :

```
10 REM PUISSANCES DE 2
20 FOR P=1 TO 10
30   PRINT "2 PUISSANCE";P;"EGALE";2^P
40 NEXT P
RUN
```

La ligne 2Ø signifie : "POUR P de 1 à 1Ø, faire ce qui suit jusqu'à l'instruction NEXT P" (NEXT = suivant).
La ligne 4Ø signifie : "à la valeur suivante de P ! (sous réserve qu'elle soit encore inférieure ou égale à 1Ø)". Cette ligne 4Ø remplace les deux lignes 4Ø et 5Ø du programme précédent écrit avec IF ... THEN. A la fin de cette ligne P augmente donc de 1.
La ligne 3Ø repésente le "corps" de la boucle, on le décale souvent de quelques caractères vers la droite pour mieux le distinguer.

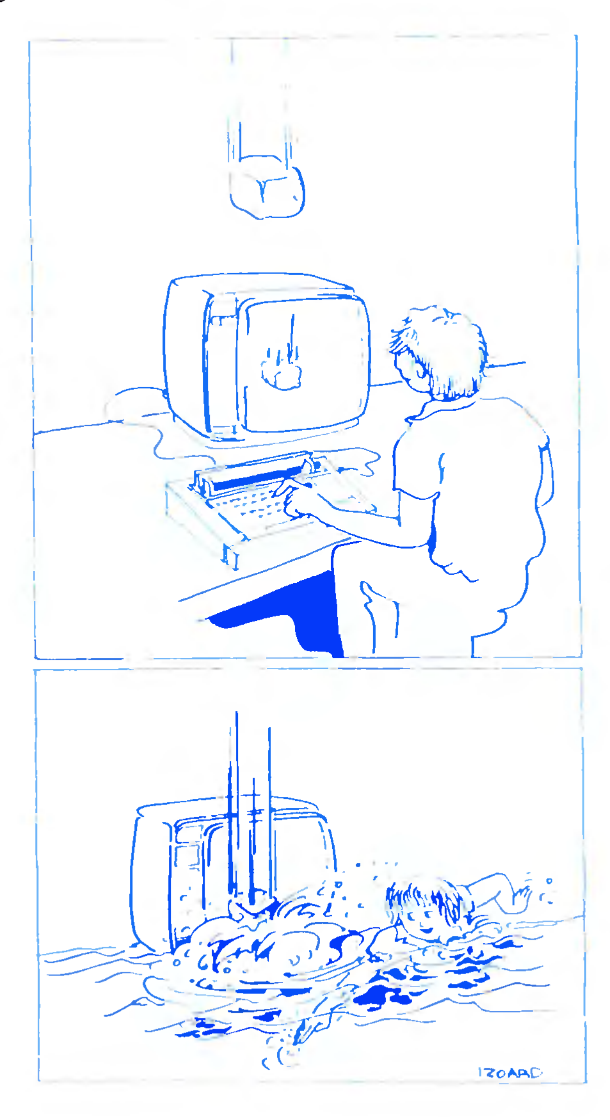
Le programme effectue 1Ø boucles et à la fin de la 1Ø^e boucle, la valeur de P est donc 11. La condition P< = 1Ø n'est plus réalisée et le programme continue à la ligne suivante.
Vous pouvez vérifier cette valeur de P en tapant, en mode direct :

```
PRINT P
11
```

La réponse est bien 11.

Un pavé dans la mare....

Qu'un projectile tombe, qu'il soit lancé vers le haut ou de droite à gauche, le programme sera presque le même. Seule l'interprétation change.



Cherchons à faire tomber verticalement à l'écran un pavé de la taille d'un caractère, c'est-à-dire un carré de 8×8. Le principe est celui des bandes lumineuses sur lesquelles "courent" des annonces : les ampoules s'allument puis s'éteignent successivement, dessinant chacune leur tour le même motif.

Le pavé est donc allumé d'abord en haut de l'écran puis effacé. Ensuite il est allumé une ligne au-dessous, effacé .... et ainsi de suite. Faisons descendre le pavé entre les colonnes 16Ø et 167. On peut ici calculer à l'avance combien de fois l'opération doit être réalisée : il y a 2ØØ lignes (de Ø à 199) et le pavé comporte 8 lignes. Le bord supérieur du pavé, qui sera notre variable LI, va donc de la ligne Ø à la ligne 192.

Essayez de rédiger le programme avec l'instruction FOR ... NEXT.

```
10 REM PAVE TOMBANT
20 CLS
30 FOR LI=0 TO 192
40   BOXF (160,LI)-(167,LI+7),1
50   CLS
60 NEXT LI
RUN
```

Vous devez voir un pavé tomber sauf si votre écran est rouge, auquel cas il est difficile de distinguer un pavé rouge sur fond rouge !

A plus grands pas

Pour faire tomber le pavé plus rapidement, il faudrait le faire descendre de plusieurs lignes entre deux affichages successifs. Dans le programme précédent la variable LI prend toutes les valeurs entre Ø et 192.

On peut faire varier LI de 8 en 8, par exemple en ajoutant dans l'instruction FOR ... NEXT un pas de variation (en anglais : STEP):

```
3Ø FOR LI = Ø TO 192 STEP 8
```

Cette ligne signifie que LI prendra successivement les valeurs de 8 en 8 à partir de Ø : Ø,8,16,24 ... 192.
Essayez cette nouvelle version et comparez-la avec celle qu'on obtient en prenant un pas de 2 ou 4 lignes.

En remontant

Maintenant que le pavé est en bas, on pourrait le faire remonter. Il faut alors que LI diminue de 192 à Ø : cela correspond à un pas négatif. L'instruction FOR ... NEXT accepte les pas négatifs :

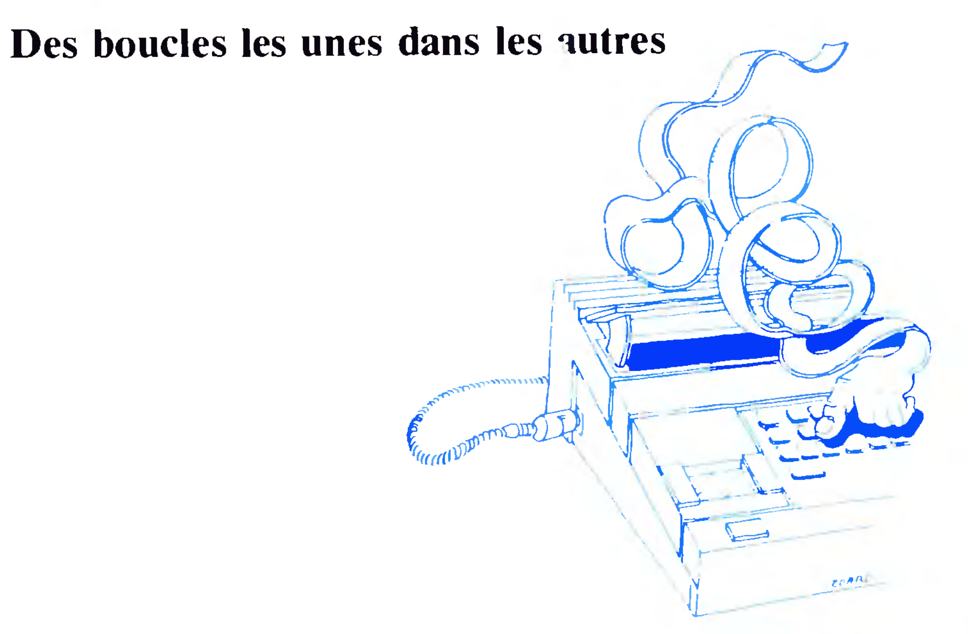
version lente

```
10 REM PAVE MONTANT
20 CLS
30 FOR LI=192 TO 0 STEP -1
40    BOXF (160,LI)-(167,LI+7),1
50    CLS
60 NEXT LI
```

version rapide

```
3Ø FOR LI = 192 TO Ø STEP - 8
```

Des boucles les unes dans les autres



Regardez le programme suivant, son architecture doit se lire "à l'œil" :

```
10 REM BOUCLES IMBRIQUEES
20 FOR A=1 TO 4
30    PRINT "A =";A
40    FOR B=1 TO 2
50       PRINT "    B =";B
60    NEXT B
70    PRINT "***"
80 NEXT A
```

Essayez d'écrire ce qui va apparaître à l'écran lorsque vous exécuterez le programme.

Puis tapez le programme et comparez avec votre prévision.

```
A = 1
    B = 1
    B = 2
***
A = 2
    B = 1
    B = 2
***
A = 3
    B = 1
    B = 2
***
A = 4
    B = 1
    B = 2
***
```

La boucle qui affiche B est exécutée pour chaque valeur de A, donc ici quatre fois. On dit que les deux boucles sont imbriquées.
La deuxième boucle doit être complètement à l'intérieur de la première : si par inadvertance vous inversez les instructions NEXT A et NEXT B, le programme ne peut pas tourner.

Un chronomètre

Il n'y a pas d'horloge sur l'ordinateur. Nous allons essayer de voir comment y remédier.



Pour faire une horloge, il faut premièrement compter le temps et deuxièmement réaliser l'affichage à l'écran.

Réservons le problème de l'affichage pour le huitième des exemples traités à la fin du livre. Nous allons donc réaliser d'abord un chronomètre, par exemple au 1/1Ø[e] de seconde.

Pour mesurer le temps, il suffit de faire faire à l'ordinateur la même opération un certain nombre de fois de suite et chronométrer avec sa montre le temps nécessaire pour cela.

L'opération la plus simple est de compter les moutons : il ne risque pas de s'endormir. Regardez ces deux lignes :

```
5Ø FOR W = 1 TO 1ØØØ
6Ø NEXT W
```

C'est une boucle qui ne comporte pas de corps, elle est équivalente à :

```
15Ø W = 1
16Ø W = W + 1
17Ø IF W < = 1ØØØ THEN 16Ø
```

C'est donc bien un compteur à l'état pur : on appelle cela une "boucle de temporisation". Elle sert à attendre plus ou moins longtemps.

Tapez :

NEW

les lignes 5Ø et 6Ø

RUN

Rien ne se passe : cependant vous constatez que le message OK n'arrive qu'au bout de quelques secondes. C'est le temps mis par l'ordinateur pour compter de 1 à 1ØØØ.
Vous pouvez vérifier qu'il s'est bien passé quelque chose en demandant en mode direct :

```
PRINT W
1ØØ1
```

Entre l'instant où vous avez appuyé sur la touche [ENTREE] et l'instant où s'est affiché OK, l'ordinateur a compté de 1 à 1ØØØ. Cela a été très vite.

Si nous voulons un chronomètre au 1/1Ø[e] de seconde, il faut afficher Ø puis Ø.1, Ø.2, Ø.3, ... et entre chaque affichage doit s'écouler exactement 1/1Ø[e] de seconde.

Essayons avec une boucle de temporisation de Ø à 1ØØ :

```
10 REM CHRONOMETRE
20 CLS
30 T = 0
40 LOCATE 15,10 : PRINT T
50 FOR W = 1 TO 100
60 NEXT W
70 T = T + 0.1
80 GOTO 40
RUN
```

Vous voyez les deux points au milieu de la ligne 4Ø ?
Ils permettent de placer deux instructions à la suite sur la même ligne.

L'affichage commence et aussitôt vous entendez l'ordinateur compter. Comparez avec votre montre : il retarde. Au bout de 1Ø secondes il indique 4.1.
Pour l'arrêter, faites CNT C.
Il faut donc remplacer 1ØØ par 1ØØ* (4.1/1Ø) = 41 :

50 FOR W = 1 TO 41

Vérifiez l'exactitude de votre chronomètre et affinez sa précision en comparant l'ordinateur à votre montre sur une plus longue durée.

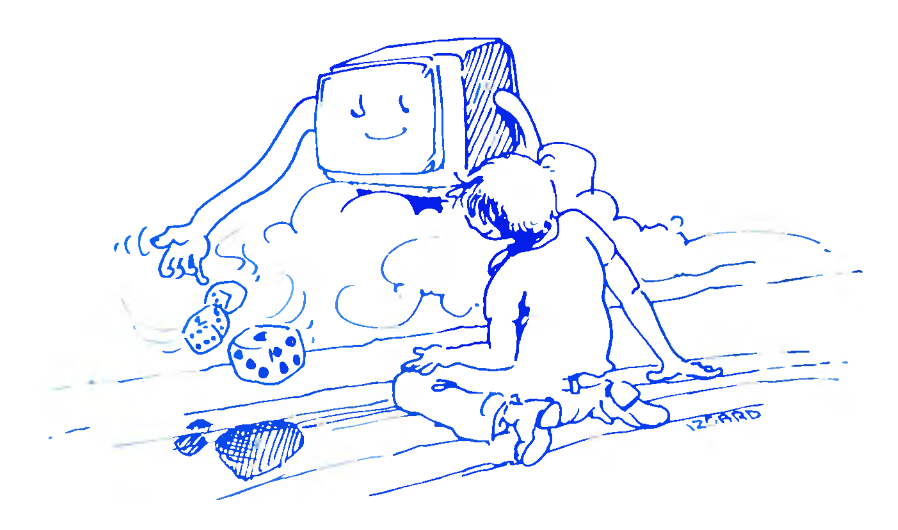
Vous avez pu constater qu'à partir de 7 s environ, votre ordinateur affiche cinq ou six décimales avec des 9. Ne vous étonnez pas. L'ordinateur calcule bien, mais avec une certaine précision (ici elle est de six chiffres) et pour lui, entre 7.299999 et 7.3, il n'y a qu'une différence d'arrondi.
Si vous cuisez vos œufs dans votre cuisine et voulez voir votre chronomètre sur votre téléviseur situé dans la pièce voisine, rajoutez cette ligne :
25 ATTRB 1, 1

L'instruction ATTRB (pour "attribut" de l'écran) permet de doubler les caractères en hauteur et en largeur.
Essayez aussi :
25 ATTRB Ø,1
25 ATTRB 1,Ø
25 ATTRB Ø,Ø
et choisissez la version que vous préférez.



Le chronomètre peut vous servir à mesurer une action en plusieurs étapes, par exemple les "coups" successifs d'un même joueur : lancez le chronomètre au début du premier coup, arrêtez-le (momentanément) avec la touche STOP, et reprenez le compte en appuyant sur une touche quelconque du clavier.

Chapitre XII/Le hasard fait bien les choses



Nous avons vu au cours des chapitres précédents que l'ordinateur traite les problèmes avec logique et rigueur. Comment pourrons-nous alors programmer un jeu de dés ou de poker ? Il faudrait pour cela pouvoir introduire un peu de hasard.

Il existe en BASIC une fonction qui s'appelle RND (de "random" : hasard) dont le rôle est de fournir des nombres tirés au hasard.

Essayons-la en mode direct :

?RND

L'ordinateur affiche :

.594972

Recommençons :

?RND

.512679

Le nombre obtenu n'est pas le même. En continuant ainsi plusieurs fois nous obtenons des nombres tous différents (compris entre Ø et 1) et apparemment tirés au hasard. On appelle cela une suite de nombres aléatoires.

Remarque : avec votre ordinateur vous n'obtiendrez peut-être pas la même suite de nombres, c'est normal. Mais ils seront tous différents et aléatoires.

Pour bien vous en assurer, écrivez ce petit programme et faites-le fonctionner :

```
10 FOR J=1 TO 20
20    PRINT RND
30 NEXT J
```

Vous constatez que tous les nombres sont compris entre Ø et 1 (sans atteindre 1) et que leurs valeurs sont très différentes. En particulier, les plus petits sont affichés en notation scientifique avec E – 2 ou E – 3.

Le tirage du loto

Nous allons utiliser cette fonction RND pour tirer les résultats du LOTO.
Le problème se pose de la manière suivante : tirer six nombres plus un (dit complémentaire), compris entre 1 et 49 sans qu'il y ait de relation entre eux.

La fonction RND nous donne un résultat compris entre Ø et 1.
Multiplions-le par 49 : 49*RND.
Modifions le programme précédent pour voir quelle suite de nombres donne ce produit :

```
10 FOR J=1 TO 20
20   PRINT 49*RND
30 NEXT J
```

Les nombres varient entre Ø et 49 (non compris)

Il faudrait supprimer les décimales pour obtenir une suite de nombres entiers. C'est le rôle de la fonction INT (de l'anglais INTEGER : entier).
Essayez-la en mode direct :

```
?INT(2.5)
2
?INT(.23)
Ø
```

Modifiez encore la ligne 2Ø du programme :

```
2Ø PRINT INT(49*RND)
```

Nous y sommes presque. Mais il reste encore un petit problème : les nombres varient entre Ø et 48 au lieu de varier entre 1 et 49.
Ajoutons donc 1 à INT(49*RND) :

```
2Ø PRINT 1 + INT(49*RND)
```

Cette fois nous obtenons bien une suite de nombres compris entre 1 et 49.

Nous pouvons maintenant rédiger notre programme :

```
10 REM LOTO
100 PRINT "NUMEROS GAGNANTS"
110 FOR I=1 TO 6
120    PRINT 1+INT(49*RND)
130 NEXT I
140 PRINT "NUMERO COMPLEMENTAIRE"
150 PRINT 1+INT(49*RND)
```

Essayez-le plusieurs fois. Il donne bien des nombres aléatoires entre 1 et 49. Mais d'une exécution à l'autre, les sept nombres tirés sont les mêmes.
En effet la fonction RND génère toujours la même suite de nombres au hasard.

Pour que deux tirages successifs du LOTO ne donnent pas les mêmes résultats, voici une solution à insérer en début de programme :

```
20 PRINT"APPUYEZ SUR UNE TOUCHE"
30 A$=INKEY$
40 X=RND
50 IF A$="" THEN GOTO 30
```

Lors de l'appui sur une touche, le tirage est à un nombre quelconque, aléatoire, et les résultats sont donc différents à chaque fois.

Voir aussi l'exemple 9 à la fin de cet ouvrage.

Un peu de graphique aléatoire

En employant la même méthode que pour le LOTO, nous allons remplir l'écran de points lumineux.
Commençons par faire la nuit :

SCREEN 7, Ø, Ø

Il va falloir demander à l'ordinateur d'allumer des points au

hasard sur l'écran. Chaque point est repéré par un numéro de colonne et un numéro de ligne. Nous allons donc tirer au hasard ces numéros de colonne et de ligne.
Le numéro de colonne est compris entre Ø et 319, il sera déterminé par :
INT(32Ø*RND)
De même la ligne, comprise entre Ø et 199 sera déterminée par :
INT(2ØØ*RND)

Il ne reste qu'à boucler pour obtenir plusieurs points :

```
10 REM ETOILES ALEATOIRES
20 CLS
30 COL = INT(320*RND)
40 LIG = INT(200*RND)
50 PSET (COL,LIG)
70 GOTO 30
```

A l'exécution vous voyez l'écran se peupler de points lumineux. Petit à petit, tout l'écran se remplit. Vous pouvez remarquer qu'il n'y a pas de région beaucoup plus claire ou plus foncée qu'une autre. C'est la preuve que la distribution des points se fait bien au hasard.
Arrêtez par CNT C.

Ce serait encore plus beau si les points étaient de toutes les couleurs. Choisissons donc la couleur de chaque point au hasard avec la fonction RND :

```
5Ø COULEUR = INT(8*RND)
6Ø PSET (COL,LIG),COULEUR
```

En poursuivant la même idée, vous pouvez vous livrer à d'autres créations aléatoires. Par exemple en traçant des lignes à la place des points :

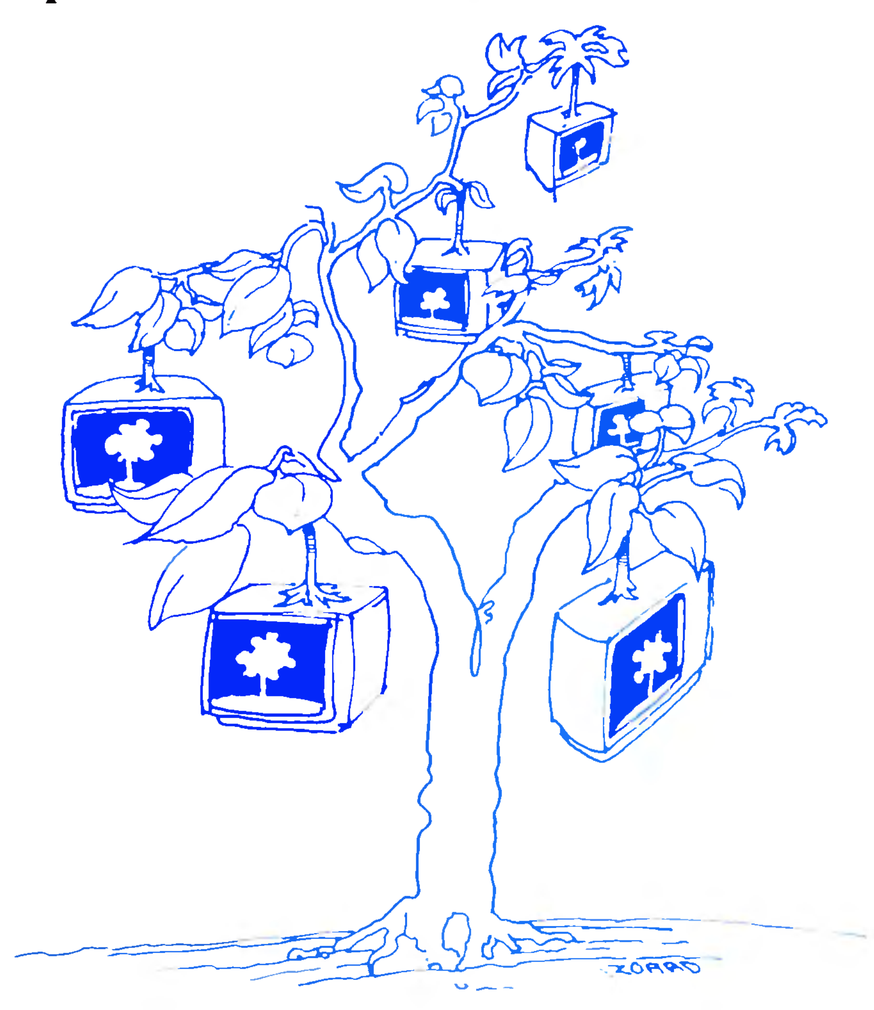
```
6Ø LINE -(COL,LIG),COULEUR
```

ou encore des boîtes dont les coins sont tirés au hasard. Mais attention : dans ce cas l'écran se remplit trop vite et devient très flou. A vous d'ajouter des règles qui limitent le hasard !

En traçant une grande quantité de lignes au hasard avec des couleurs tirées au hasard, vous avez pu constater que les couleurs se chevauchent un peu. Quelquefois, un petit segment horizontal prend la couleur de la ligne qui le traverse.
Il ne s'agit pas d'une erreur de programmation mais d'une particularité du dispositif de visualisation.

Nous avons vu, au moment d'effacer un point ou une ligne, que chaque point de l'écran a deux états possibles :
— ou il appartient au fond de l'écran
— ou il appartient à un dessin, à une forme.
Or le mécanisme de visualisation des couleurs attribue les couleurs par groupe de huit points horizontaux consécutifs. Ainsi, une ligne horizontale de 320 points est-elle découpée en 40 segments de huit points. Sur chaque segment, il ne peut y avoir qu'une couleur pour le fond et une couleur pour la forme. Et quand on vient écrire sur un segment un point d'une couleur donnée, tous les points déjà marqués prennent cette couleur.
Ça n'est pas très grave, mais il est utile de le savoir pour éviter à la couleur d'un dessin de venir mordre sur un dessin très voisin.

Chapitre XIII/Une forêt de sous-programmes



Dessine-moi un arbre

Nous allons essayer de dessiner un arbre sur l'écran. Il sera un peu symbolique au début, mais il ne tiendra qu'à vous d'en améliorer l'allure ultérieurement.

Donnons-lui la forme d'un triangle posé sur un pied ⍋. Si nous partons du point situé en bas et à gauche et si ce point est situé sur la colonne C et la ligne L, la partie de programme qui dessine cet arbre s'écrit de la manière suivante :

```
1000   ARBRE
1010 LINE (C,L)-(C+20,L)
1020 LINE (C+10,L)-(C+10,L-20)
1030 LINE (C,L-20)-(C+20,L-20)
1040 LINE -(C+10,L-40)
1050 LINE -(C,L-20)
```

Passons sur le détail des lignes. Recopiez-le, vous verrez bien après.
Complétons ce programme.
D'abord, il faut effacer l'écran :

```
2Ø CLS
```

Puis situer le point d'où l'on part ; par exemple en (1ØØ, 1ØØ) :

```
3Ø C = 1ØØ : L = 1ØØ
```

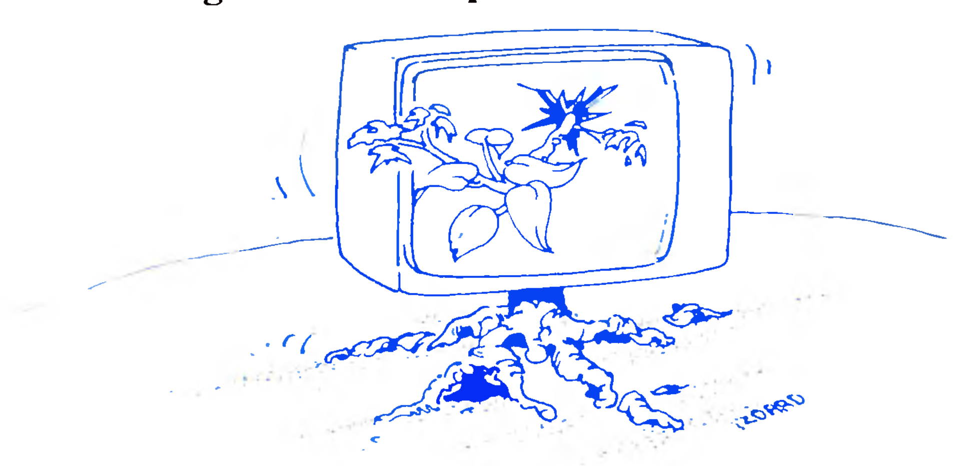
Cette fois le programme est complet.

```
RUN
```

C'est bon. Vous voyez à l'écran, quelque chose qui ressemble à ce que l'on voulait.

Nous avons choisi des numéros de lignes qui ne se suivent pas ; c'est que nous avons l'idée d'en rajouter quelques-unes. De toute façon, l'ordinateur ne fait pas d'histoires, il les exécute dans l'ordre croissant même si les numéros sont très éloignés. (J'espère que vous n'avez pas cru d'après ce qui précède que les programmes ne marchent qu'avec des numéros de 1Ø en 1Ø, ce n'est qu'une vieille habitude). C'est d'ailleurs le moment d'apporter une précision, le numéro d'une ligne ne peut pas dépasser 63999. Allez donc savoir pourquoi !

Pour changer l'arbre de place



Les cinq lignes qui tracent le dessin, de 1Ø1Ø à 1Ø5Ø sont toutes calculées à partir du point situé en C et L. Pour changer notre arbre de place, il nous suffit donc de changer la ligne 3Ø. Pour le mettre plus à gauche :

```
3Ø C = 16Ø : L = 1ØØ
```

La partie de programme qui dessine est indépendante de la position du dessin sur l'écran. Ceci va nous servir par la suite.

J'en dessinerais bien un deuxième

Si nous voulons dessiner un deuxième arbre identique au premier, mais à côté, faudra-t-il recopier les lignes 1Ø1Ø, 1Ø2Ø, 1Ø4Ø et 1Ø5Ø? Non, parce que nous allons en faire un sous-programme. Un sous-programme est une partie de programme qui peut-être utilisée plusieurs fois de façon indépendante.

Opérons la transformation :
Le sous-programme, qui va commencer en ligne 1ØØØ, doit se terminer par une instruction de retour :

```
1Ø6Ø RETURN
```

Il peut commencer par une instruction quelconque, mais il est très souhaitable de mettre en première ligne un commentaire qui indique ce qu'il fait : ici c'est "arbre".
L'appel du sous-programme s'écrit alors :

```
4Ø GOSUB 1ØØØ
```

Sous-programme se dit en anglais "subroutine", d'où le nom de l'instruction GOSUB (GO to SUBroutine : aller au sous-programme). Jusque-là, nous n'avons fait qu'une seule chose, modifier la forme du programme, mais pas ce qu'il fait :

```
RUN
```

Le dessin est toujours au même endroit, mais il apparaît un message d'erreur :

```
?RG Error in 1Ø6Ø
```

En effet, l'ordinateur n'aime pas les mélanges entre programmes et sous-programmes. Il faut mettre une instruction à la fin du programme qui fasse la séparation et indique où s'arrête le programme principal :

```
1ØØ END
```

Vérifiez avec une nouvelle exécution. Plus d'erreur, c'est bien ça. Revenons à ce qui nous intéresse, dessiner un deuxième arbre. Il suffit maintenant d'en fixer le point de départ :

```
5Ø C = 13Ø : L = 1ØØ
```

et d'appeler le sous-programme :

```
6Ø GOSUB 1ØØØ
```

Refaites une exécution. Vous en voyez bien deux.

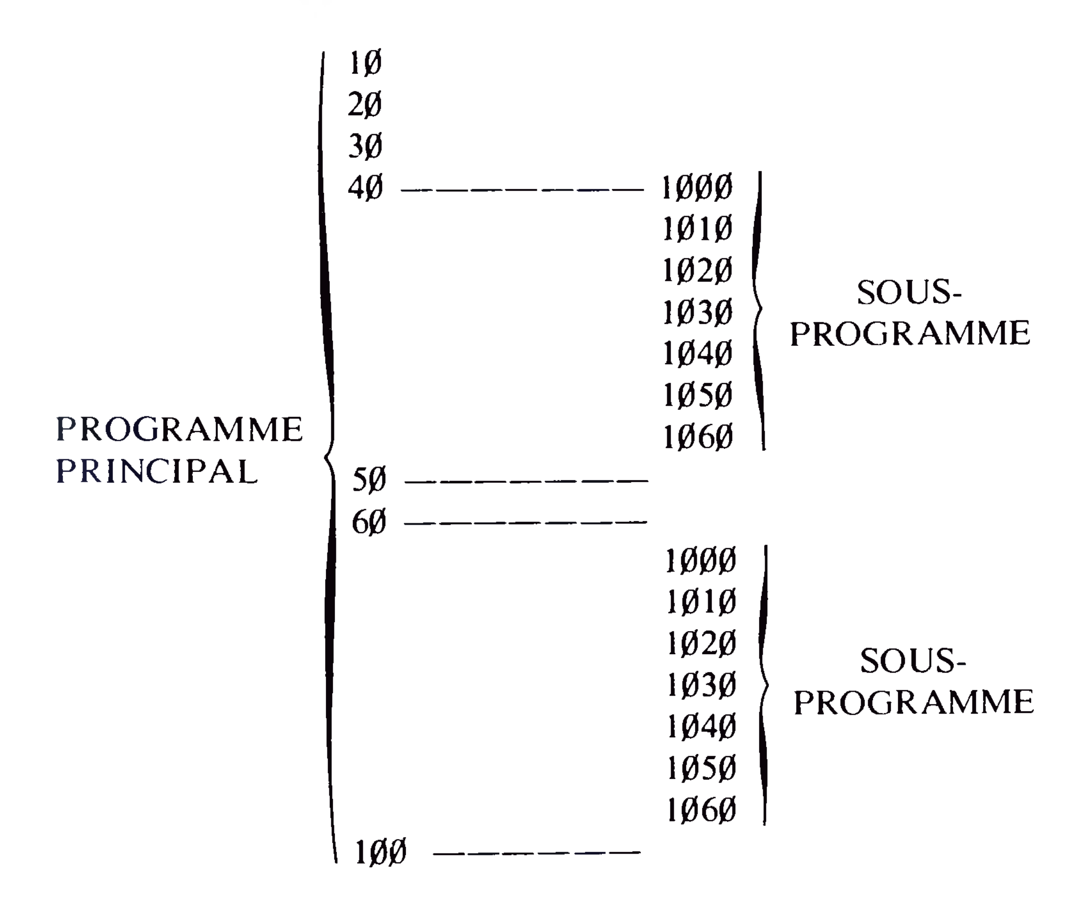
Il est temps maintenant d'examiner en détail le fonctionnement du sous-programme.

Nous avons vu que l'ordinateur exécute un programme dans

l'ordre des numéros de lignes croissants. Il lui arrive de ne pas respecter cet ordre, lorsqu'il rencontre une instruction GOTO (branchement dit "inconditionnel") ou IF ... THEN ... (branchement dit "conditionnel").

Lors d'une instruction GOSUB, l'ordinateur va effectivement à la ligne indiquée, mais il se souvient de l'endroit d'où il est parti. Et lors de l'instruction RETURN, à la fin du sous-programme, il revient à l'instruction qui suit le GOSUB.

Dans le programme que nous venons d'écrire, les lignes sont exécutées dans l'ordre suivant :



Un programme qui laisse des traces : TRON, TROFF

Il existe un moyen simple de vérifier ce que nous venons d'aborder au-dessus. C'est une commande qui inscrit sur l'écran les numéros des lignes au fur et à mesure de leur exécution : TRON (pour TRACE ON).

Faites :

TRON suivi de :

RUN

Voici le résultat :

```
[30][40][1000][1010][1020][1030][1040][1
050][1060][50][60][1000][1010][1020][103
0][1040][1050][1060][70]
```

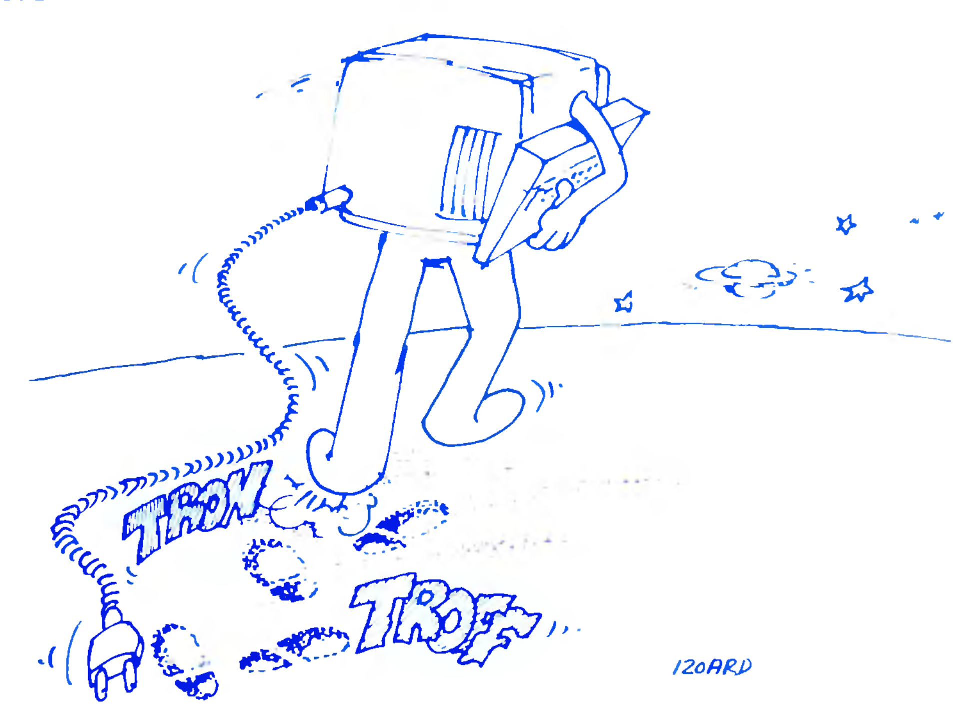
ce qui confirme bien ce que nous venons de dire. Cette commande est très utile lorsqu'on veut lever le doute sur le fonctionnement d'un programme, car elle permet d'en suivre l'exécution ligne par ligne.

Mais vous ne souhaitez peut-être pas que son effet agisse indéfiniment. Pour y mettre fin, il existe la commande opposée TROFF (pour TRACE OFF).

Faites :

TROFF puis :

RUN. Tout est redevenu normal.



Varier un peu le paysage

Dessiner toujours la même chose peut vous sembler un peu monotone. Nous allons introduire une nouvelle variété.

Un arbuste par exemple :

Faisons directement le sous-programme :

```
2000   ARBUSTE
2010 LINE (C,L)-(C+10,L)
2020 LINE (C+5,L)-(C+5,L-10)
2030 LINE (C,L-10)-(C+10,L-10)
2040 LINE -(C+5,L-20)
2050 LINE -(C,L-10)
2060 RETURN
```

Il ressemble assez à un arbre, mais en plus court. Pour le voir à l'écran, remplacez le deuxième arbre par un arbuste :

```
60 GOSUB 2000
```

Faites RUN : ça change tout de même un peu. Libre à vous maintenant d'inventer de nouvelles formes que vous écrirez en 3000, 4000, 5000, ... etc.; il vous suffira pour les dessiner d'appeler le sous-programme correspondant.

Nous allons essayer d'améliorer encore le fonctionnement de notre programme de la façon suivante :

Proposons à l'utilisateur de choisir lui-même le dessin. Il suffit pour cela de lui poser la question et suivant sa réponse d'aller à l'un ou l'autre des sous-programmes. Il existe en BASIC une instruction qui va nous faciliter la tâche : ON ... GOSUB.

Voici son utilisation :

```
10 ' CHOIX DU DESSIN
20 CLS
30 C=100 : L=100
40 PRINT "VOULEZ-VOUS"
50 PRINT "1-UN ARBRE, 2-UN ARBUSTE";
60 INPUT REPONSE
70 ON REPONSE GOSUB 1000,2000
100 END
```

La suite du programme est identique.

Essayez-le : suivant que vous répondez 1 ou 2 à la question, le programme dessine un arbre ou un arbuste. Par contre, si vous répondez par un autre nombre, il ne dessine rien.

Le fonctionnement de
ON REPONSE GOSUB 1000, 2000
est le suivant :

Si REPONSE est égal à 1, alors aller au sous-programme 1ØØØ
Si REPONSE est égal à 2, alors aller au sous-programme 2ØØØ
Si REPONSE est égal à une autre valeur, passer à la suite.

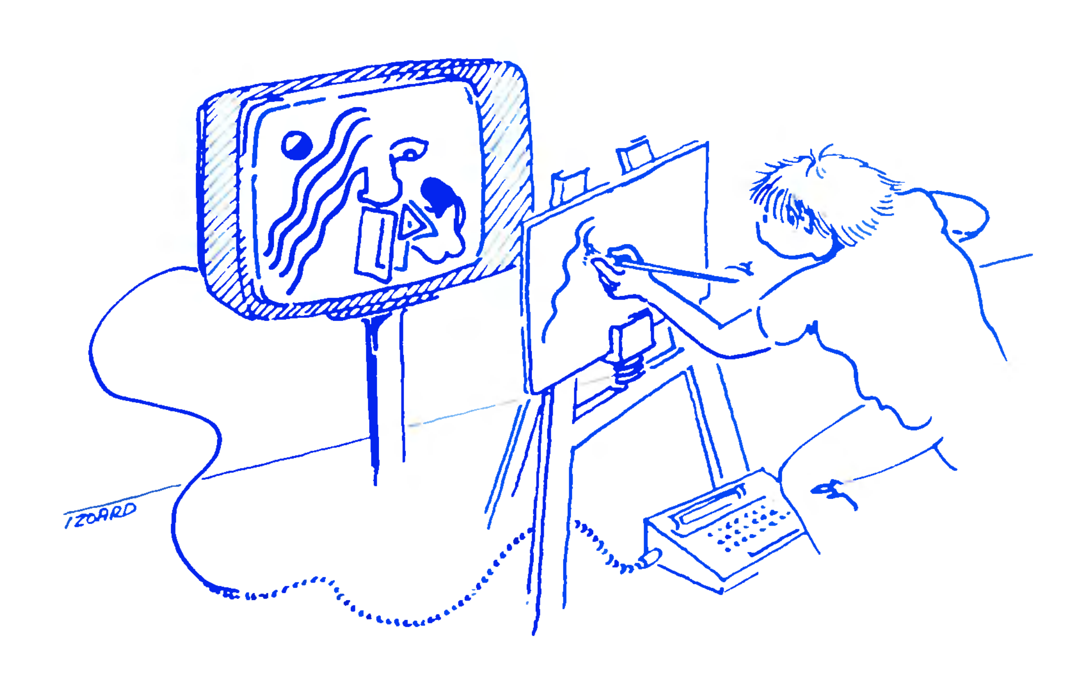
Cette instruction est pratique, car elle permet de rassembler en une seule ligne ce qui aurait été plus long à écrire avec IF ... THEN.

De manière générale, cette instruction peut-être employée avec un nombre quelconque de sous-programmes. Par exemple :

ON REPONSE GOSUB 1ØØØ, 2ØØØ, 3ØØØ, 4ØØØ, 5ØØØ.

Dans ce cas, suivant que REPONSE vaut 1, 2, 3, 4, ou 5, le sous-programme en 1ØØØ, 2ØØØ, 3ØØØ, 4ØØØ, ou 5ØØØ sera exécuté. Et si REPONSE a une valeur différente, l'exécution continuera à la suite.

Encore une précision : supposons que vous ayez répondu 1.5 à la question. La variable REPONSE vaut alors 1.5.
A quel sous-programme aller ? Et bien en 2ØØØ.
En effet, l'ordinateur arrondit le nombre qui suit ON à l'entier le plus proche avant de choisir le sous-programme correspondant.



Chapitre XIV/Des caractères en chaînes

Qu'est-ce qu'une chaîne de caractères

Vous avez déjà utilisé des chaînes de caractères, comme Monsieur JOURDAIN (1) faisait de la prose : sans le savoir.

Lorsque vous écrivez :

PRINT "*BONJOUR*"

Ce qui est entre guillemets, soit *BONJOUR* est une chaîne de caractères, c'est-à-dire une suite d'au maximum 255 caractères. Seul le guillemet est interdit au milieu d'une chaîne, car il sert justement à limiter les chaînes.

Jusqu'ici rien de nouveau !

"ZAZIE"
"999"
"5 + 7 ="
"Clémence JOLY - 3 Rue des Plantes - PARIS"
sont des chaînes de caractères.

Jusqu'à maintenant, vous les avez utilisées seulement dans l'instruction PRINT. Vous allez voir qu'il existe un grand nombre d'autres possibilités.



(1) Monsieur JOURDAIN, personnage principal de la comédie de Molière "Le Bourgeois Gentilhomme" aurait certainement fait de l'informatique sans le savoir, s'il avait vécu notre époque.

Par exemple, si vous désirez garder la liste de vos amis avec leurs numéros de téléphone, il va falloir les ranger quelque part dans la mémoire de l'ordinateur. Chaque nom et chaque numéro de téléphone formeront une chaîne de caractères et chaque chaîne sera repérée dans la mémoire par un nom qui la désigne.

C'est la même chose que pour ranger des nombres en mémoire : on les affecte à des variables de noms différents.
La seule différence est que le nom d'une variable chaîne de caractères (on dit aussi variable alphanumérique) doit se terminer par $.

A$, N1$, REPONSE$ sont des noms possibles de variables chaînes de caractères.

```
10 N1$ = "Clemence JOLY"
20 T1$ = "(98) 666-66-66"
30 PRINT N1$,T1$
```

La ligne 1∅ affecte à la variable N1$ la chaîne "Clémence JOLY"
La ligne 2∅ affecte à la variable T1$ la chaîne "(98) 666 – 66 – 66"

Si dans la suite du programme vous voulez réutiliser ces chaînes, il suffit de les appeler par leurs noms : c'est beaucoup plus rapide.

En outre, si un numéro de téléphone change, il suffit de le changer une fois pour toutes, dans l'écriture de la variable correspondante. Partout où cette variable apparaîtra ensuite, elle donnera le bon numéro.

On ne risque pas de confondre une variable chaîne (ou alphanumérique) avec une variable numérique puisque la première doit se terminer par $.

Ainsi :

```
A = 5
A$ = "TO7"
```

définissent 2 variables A et A$ de types différents.

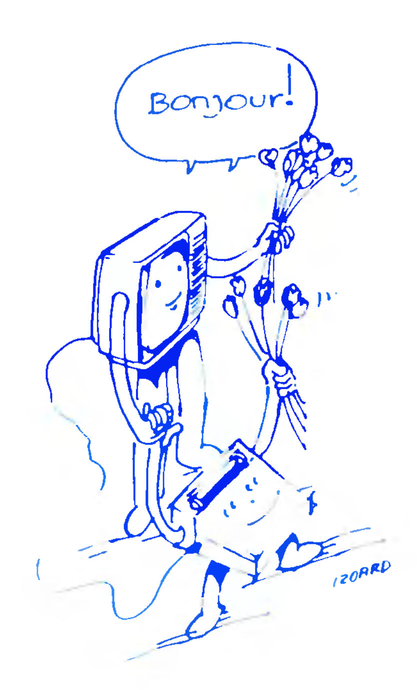
On le vérifie facilement par :

```
PRINT A;A$
```

L'ordinateur poli

Si vous avez déjà vu des démonstrations sur ordinateur, vous avez sûrement constaté qu'en général, ils sont très polis :

Ils commencent par vous demander votre nom, puis vous disent bonjour ! Nous ne pouvons pas faire moins.



Pour entrer un nom, c'est-à-dire une chaîne de caractères en mémoire, BASIC utilise la même instruction que pour entrer des nombres : INPUT.

Essayez le petit programme suivant :

```
NEW
10 PRINT "QUEL EST VOTRE NOM ?"
20 INPUT NOM$
30 PRINT "BONJOUR"
40 PRINT NOM$
RUN
```

Lorsque l'ordinateur voit INPUT, il affiche un point d'interrogation et attend votre réponse.

Vous pouvez taper :

```
JULES
```

Il vous répondra :

```
BONJOUR
JULES
```

Pour que la variable NOM$ s'affiche à la suite de BONJOUR, il suffit de la mettre en fin de ligne 3Ø, séparée par un point virgule :

```
3Ø PRINT "BONJOUR" ; NOM$
```

Effacez la ligne 4Ø.

Mais les 2 chaînes sont accolées l'une à l'autre. En introduisant un espace à la fin de "BONJOUR", on sépare les 2 mots :

```
30 PRINT "BONJOUR " ; NOM$
```

Vous pouvez aussi regrouper les 2 premières instructions dans le même INPUT :

```
10 INPUT "QUEL EST VOTRE NOM" ; NOM$
30PRINT "BONJOUR " ; NOM$
```

Quelques notes de musique

Grâce au synthétiseur, vous allez pouvoir vous essayer à la composition. Vous pourrez ainsi composer des airs, sonoriser vos jeux vidéo, utiliser des indicatifs musicaux pour séparer les différentes parties d'un programme, etc.

Qui dit musique, dit notes. Elles seront appelées par leur nom sauf SOL qui est raccourci en SO :
DO RE MI FA SO LA SI
et même P pour la pause.
Ce sont donc des chaînes de caractères.

Pour jouer une note, il suffit de donner l'ordre PLAY ! ("joue").

```
10 PLAY "DO"
```

Si vous voulez qu'il répète un peu, ajoutez :

```
20 GOTO 10
```

Vous vous souvenez que pour arrêter temporairement, il suffit d'appuyer sur la touche [STOP]. Pour jouer plusieurs notes à la suite, on les écrit les unes à la suite des autres à l'intérieur des guillemets :

```
10 PLAY "DOREMIFASOLASI"
```

On peut aussi écrire, de manière plus lisible, mais avec plus de frappe :

```
10 PLAY "DO;RE;MI;FA;SO;LA;SI".
```

Vous savez que les mêmes notes peuvent êtres jouées "plus haut" ou "plus bas", c'est-à-dire à un octave supérieur ou inférieur.
La machine à un beau filet de voix, car elle couvre 5 octaves. De quoi rendre jalouse la Castafiore !

L'octave le plus bas, c'est-à-dire les notes les plus basses est appelé O1 (attention lettre O et non chiffre Ø).

```
1Ø PLAY "O1DOREMIFASOLASI"
```

Changez ensuite le 1 en 2, 3, 4, 5.

Pour O4 vous retrouvez la gamme que nous avions tout à l'heure sans préciser d'octave : c'est en effet l'octave standard.



Si vous voulez jouer la gamme complète, il faut donc rajouter à la fin O5DO, étant sous-entendu que les premières notes sont dans l'octave standard.

```
1Ø PLAY "DOREMIFASOLASIO5DO"
```

Ajoutons le rythme

Le rythme est donné par la durée différente des notes.
Écoutez la différence entre ces 2 instructions :

```
1Ø PLAY "DOREMI"
2Ø PLAY "L48DOREMI"
```
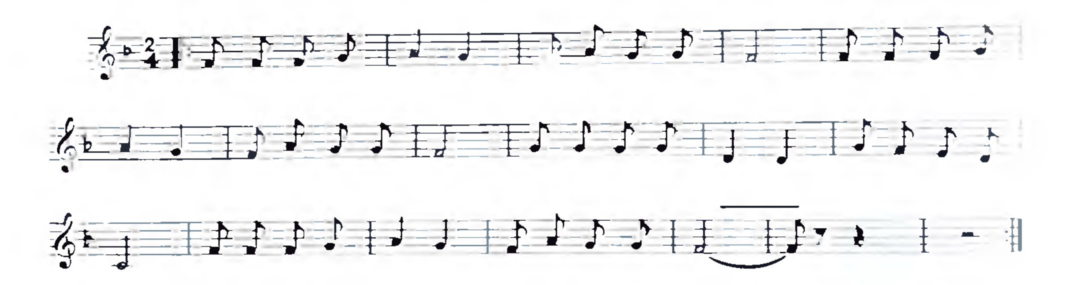
La deuxième série de notes est 2 fois plus longue que la première.
Changez 48 en 96 : la durée double encore. C'est la durée maximum.
Changez 96 en 24 : les 2 séquences ont la même durée (L24 est la durée standard.)
Changez 24 en 12 : cette fois le deuxième morceau est 2 fois plus court que le premier.

Ce qui correspond aux durées des notes suivantes :
croche L12
noire L24
blanche L48
ronde L96

Mais vous pouvez très bien utiliser des notes de durée 1Ø ou 23, si vous voulez bousculer le rythme.

Il vous reste à découvrir comment faire varier le tempo, l'attaque, ajouter des dièses, des bémols et des silences ... De quoi faire une symphonie. Mais pour les airs simples, les octaves et les durées nous suffiront.

Un air mystérieux



Essayons de jouer le petit air ci-dessus.

Il serait possible d'écrire toutes les notes les unes à la suite des autres, mais vous pouvez imaginer qu'un tel programme est difficile à lire, et à modifier.

En outre, les mêmes mesures se répètent. On peut ranger chaque mesure dans une variable :

```
1Ø A$ = "L12FAFAFASO"    (1re mesure)
2Ø B$ = "L24LASO"        (2e mesure)
3Ø C$ = "L12FALASOSO"    (3e mesure)
4Ø D$ = "L48FA"          (4e mesure)
```

Pour jouer les 4 mesures, il faut former la chaîne équivalente à la suite de toutes les notes. Pour cela, on utilise le signe + :

```
1ØØ M1$ = A$ + B$ + C$ + D$
11Ø PLAY M1$
RUN
```

Cette opération au lieu de s'appeler tout bêtement une addition, comme avec les nombres, s'appelle ici une "concaténation".

Si vous voulez vérifier le résultat, faites en mode direct : PRINT M1$.

Revenons à notre air : la suite des mesures, 5, 6, 7 ,8 est identique à celle des mesures 1, 2, 3, 4. Il reste à rentrer les mesures 9, 10, 11, 12 :

```
5Ø   E$ = "L12SOSOSOSO"
6Ø   F$ = "L24RERE"
7Ø   G$ = "L12SOFAMIRE"
8Ø   H$ = "L48DO"
9Ø   M2$ = E$+F$+G$+H$
1ØØ  M1$ = A$+B$+C$+D$
11Ø  PLAY M1$+M1$
12Ø  PLAY M2$+M1$ + "L12FA"
```

Pianoter à l'écran

Si vous voulez tester rapidement une suite de notes, nous allons écrire un programme qui vous permettra de jouer directement les notes à l'écran avec le crayon optique.

Les notes sont affichées, chacune dans un rectangle, et la note correspondante est jouée lorsqu'on appuie à l'intérieur du rectangle avec le crayon optique.

DO	RE	MI	FA	SO	LA	SI

On pourrait écrire ce programme en comparant les coordonnées du point désigné à l'écran avec les numéros de colonnes et de lignes des différents rectangles. Mais une instruction permet d'éviter ce grand nombre de tests (4 par rectangle !).

Elle attribue à chaque rectangle un numéro de zone de Ø à 7 (on peut définir un nombre inférieur de zones : ici nous avons 7 zones) et suivant le rectangle visé, branche le programme à une ligne différente :

PEN Ø ; (2Ø, 8Ø) – (44, 96)
définit la zone Ø formée par le rectangle (2Ø, 8Ø) – (44, 96)

PEN 1 ; (44, 8Ø) – (68, 96)
définit la zone 1 formée par le rectangle (44, 8Ø) – (68 – 96), et ainsi de suite.

ON PEN GOTO 13Ø, 14Ø, 15Ø, 16Ø, 17Ø, 18Ø, 19Ø
branche le programme en ligne 13Ø si on appuie sur la zone Ø
branche le programme en ligne 14Ø si on appuie sur la zone 1
etc.

Voici le programme :

```
10 ' GAMME
20 CLS
30 SCREEN 0,6,6
40 LOCATE 3,11,0
50 PRINT "DO RE MI FA SO LA SI"
60 FOR I=0 TO 6
70    COL=20 + I *24
80    BOX    (COL,80)-(COL+24,96)
90    PEN I ;(COL,80)-(COL+24,96)
100 NEXT I
110 '
120 ONPEN GOTO 140,150,160,170,180,190,
    200
130 GOTO 120
140 PLAY "DO" : GOTO 120
150 PLAY "RE" : GOTO 120
160 PLAY "MI" : GOTO 120
170 PLAY "FA" : GOTO 120
180 PLAY "SO" : GOTO 120
190 PLAY "LA" : GOTO 120
200 PLAY "SI" : GOTO 120
```

RUN

Vous pouvez maintenant tester des mélodies simples qui accompagneront vos programmes.

Quelle est la structure du programme ?

— Afficher les noms des notes (L.4Ø – 5Ø). Le chiffre Ø dans l'instruction LOCATE rend le curseur invisible pendant le déroulement du programme.

— Dessiner les 7 rectangles et faire de chacun d'eux une zone numérotée par l'instruction PEN (L.6Ø à 1ØØ).
La variable choisie dans la boucle (COL) est le numéro de la colonne de gauche de chaque rectangle.
D'un rectangle à l'autre, COL augmente de 24, largeur de chaque rectangle.

— Savoir quelle est la zone désignée par le crayon optique et en déduire le branchement correspondant (L.12Ø)

— Jouer la note de musique. (L. 140 à 2ØØ).



Chapitre XV/Jouer avec les mots

Avez-vous dit oui ou non ?

Maintenant vous allez voir que l'ordinateur peut entamer le dialogue :
Tapez ce petit programme :

```
10 ' OUI OU NON
20 PRINT "VOULEZ-VOUS UN PETIT AIR ?"
30 INPUT R$
40 IF R$="NON" THEN PRINT "AU REVOIR"
50 IF R$="OUI" THEN PLAY "DOMISOLSOLSDO"
RUN
```

Le programme compare la chaîne R$, c'est-à-dire votre réponse, d'abord avec la chaîne "NON" puis avec la chaîne "OUI". Il agit ensuite en conséquence !

Pour comparer des chaînes de caractères, on utilisera les mêmes symboles que pour les nombres :
= pour "égal" ou identique
< > pour "différent de"

Faites à nouveau tourner le programme et répondez "NON MERCI" ou "OUI!" ou toute autre réponse que vous pourrez imaginer.
En ligne 4Ø l'ordinateur voit que votre réponse est différente de "NON", il poursuit donc en ligne 5Ø et voit que la réponse est encore différente de "OUI". Alors il continue encore et reste muet puisqu'il n'y a plus d'instructions.

On peut prévoir ce cas en ajoutant :

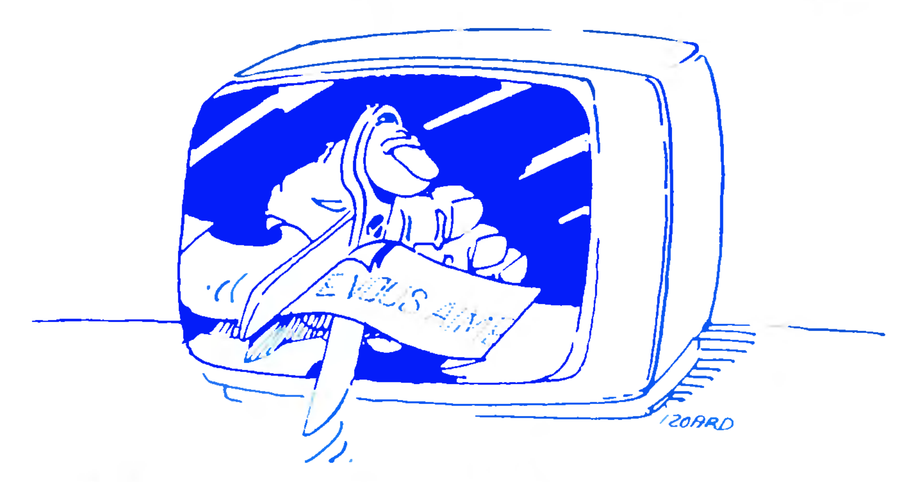
```
6Ø IF R$ < > "OUI" AND R$ < > "NON" THEN 2Ø
```

Avec ces 2 opérateurs de comparaisons pour les chaînes de caractères < > et = vous pourrez rédiger des programmes qui posent des questions sur n'importe quel sujet et dire si celui qui tape la réponse s'est trompé ou non.

Couper un mot ou une phrase en morceaux

Pour cela vous allez voir plusieurs sortes de tronçonneuses à votre disposition. La première est la fonction MID$ (pour

MIDDLE = milieu).
Elle vous coupe dans une chaîne de caractères un morceau de la longueur que vous désirez à partir du caractère que vous désirez.



Prenons par exemple, la chaîne suivante :

A$ = "MY MACHINE IS BEAUTIFUL"

Et essayez en mode direct :

```
?MID$(A$, 4, 1)
M
?MID$(A$, 4, 2)
MA
?MID$(A$, 4, 7)
MACHINE
```

Dans ce dernier exemple, la fonction MID$ a extrait 7 caractères à partir du 4e dans la chaîne A$ (n'oubliez pas que les espaces comptent !).

Vous pouvez maintenant prendre votre papier pour rédiger le programme qui va afficher la chaîne A$ verticalement à l'écran : le 1er caractère sur la 1re ligne, le 2e sur la 2e ligne ...

N'oubliez pas, que lorsque l'ordinateur doit répéter plusieurs fois la même action, il est recommandé de placer cette action dans une boucle.

```
NEW
10 ' ECRITURE VERTICALE
20 A$="MY MACHINE IS BEAUTIFUL"
30 FOR I=1 TO 23
40    PRINT MID$(A$,I,1)
50 NEXT I
RUN
```

La chaîne A$ comporte 23 caractères et le programme exécute 23 fois de suite l'impression du caractère numéro I de la chaîne A$.

Si vous voulez imprimer verticalement une autre chaîne de caractères, il faudra d'abord compter le nombre total de caractères. Ceci nécessitera à chaque fois une modification de la ligne 3Ø du programme.

Il existe une fonction qui évite de compter le nombre total de caractères d'une chaîne : LEN (length = longueur).

Essayez en mode direct :

```
? LEN (A$)
 23
? LEN ("42,5Ø F")
 7
```

En effet la chaîne "42,5Ø F" comporte 7 caractères.

Remarquez que le résultat de la fonction LEN est un nombre (le nombre de caractères de la chaîne) alors que celui de la fonction MID$ est une chaîne. C'est pour cette raison que le nom de la fonction MID$ se termine par le caractère $.

Voyez-vous comment modifier le programme précédent pour afficher verticalement une phrase quelconque ?

```
10 ' ECRITURE VERTICALE
20 INPUT"DONNEZ UN MOT OU UNE PHRASE";A$
30 FOR I=1 TO LEN (A$)
40   PRINT MID$(A$,I,1)
50 NEXT I
```

L'orthographe sans peine, ou ... le pluriel automatique

Peut-être êtes-vous brouillé avec l'orthographe ?
Alors nous allons demander à la machine de nous aider un peu.
Nous allons lui apprendre à mettre un mot au pluriel. En général, je dis bien "en général", le pluriel d'un mot se forme en ajoutant un "s" à la fin du mot.
Jusqu'ici c'est simple : appelons MOT$ le mot et PLUR$ son pluriel.

```
10 ' PLURIEL
20 INPUT "QUEL MOT" ; MOT$
110 PLUR$ = MOT$ + "S"
120 PRINT "PLURIEL : " ; PLUR$
```

Vous vous demandez sans doute pourquoi on saute de la ligne 2Ø à la ligne 11Ø. Alors testez le programme avec un certain nombre de mots :
RUE, JOURNAL, AVION, NEZ, CROIX ...
Avant d'appliquer la recette du cas général, il faut donc vérifier que le mot n'entre pas dans un cas particulier. Nous allons voir les 2 cas particuliers suivants :
— les mots en "AL" qui forment leur pluriel en "AUX"
— les mots qui finissent par S, Z ou X qui ne changent pas au pluriel, exemple : un nez, des nez.
Il restera encore le cas des mots en AU, EU, EAU qui prennent un X et les fameux bijoux, cailloux, choux, genoux, hiboux, joujoux et poux ! Vous pourrez les rajouter pour obtenir un programme (presque) complet !
La structure du programme sera donc :

2Ø Introduire le mot
3Ø Si le mot n'est pas en AL alors aller en 7Ø (sinon continuer)
   Déterminer le pluriel d'un mot en AL
   Aller en 120 pour imprimer ce pluriel.
7Ø Si le mot n'est pas en S, Z ou X alors aller en 11Ø (sinon continuer)
   Déterminer le pluriel d'un mot en S, Z ou X
   Aller en 120 pour l'imprimer.
1ØØ Déterminer le pluriel du cas général
12Ø Impression du pluriel

Comment faire pour le premier cas particulier : JOURNAL ? (ou tous les mots en "AL")
Pour former le pluriel, il faut retirer "AL" et le remplacer par "AUX".
Vous pouvez essayer de faire cela avec les 2 instructions MID$ et LEN.
C'est possible, mais il existe deux fonctions qui rendront l'écriture de ce problème beaucoup plus facile :
LEFT$ (left = gauche) et RIGHT$ (right = droite)
Essayez en mode direct :

```
?LEFT$ ("JOURNAL", 1)
J
?LEFT$ ("JOURNAL", 2)
JO
?LEFT$ ("JOURNAL", 7)
JOURNAL
```

Puis testez de même RIGHT$, en plaçant JOURNAL dans une variable (ce sera plus rapide) :

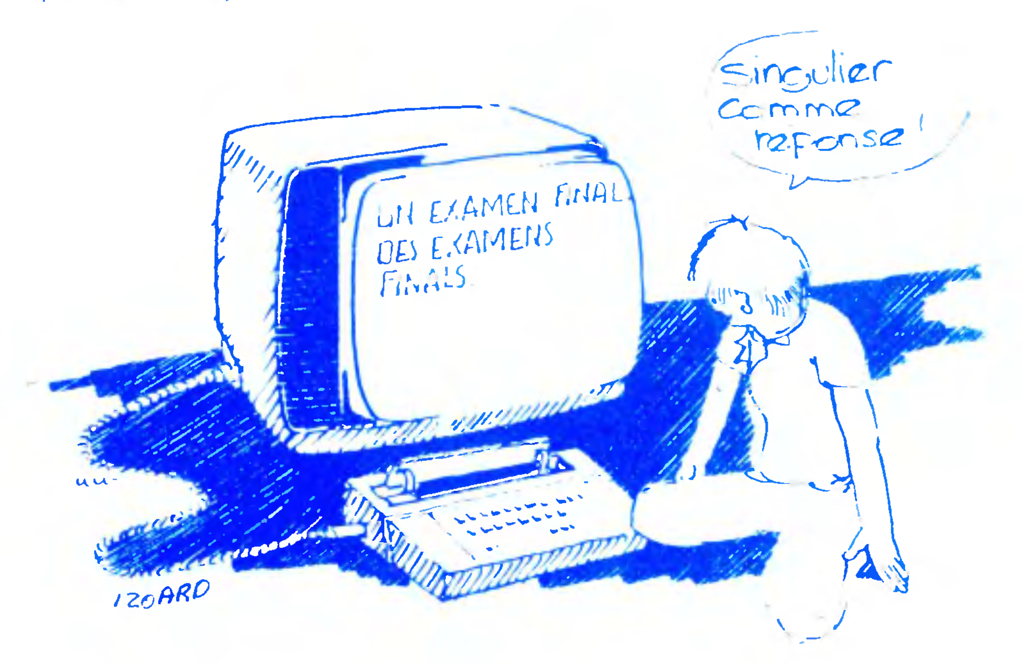
```
A$ = "JOURNAL"
?RIGHT$ (A$, 1)
L
?RIGHT$ (A$, 7)
JOURNAL
```

Ainsi LEFT$ et RIGHT$ permettent d'extraire d'une chaîne de caractères à partir de la gauche ou de la droite, le nombre de caractères que l'on veut.

Les deux dernières lettres d'un mot seront donc
RIGHT$ (MOT$, 2)
Le mot moins ses deux dernières lettres sera
LEFT$ (MOT$, L – 2), en appelant L le nombre total des lettres :
L = LEN (MOT$)

D'où l'écriture du 1er cas particulier :

```
3Ø IF RIGHT$ (MOT$, 2) < > "AL" THEN 7Ø
4Ø L = LEN (MOT$)
5Ø PLUR$ = LEFT$ (MOT$, L – 2) + "AUX"
6Ø GOTO 12Ø
```



Testez le programme : un cheval, des chevaux !

Essayez maintenant de rédiger le 2e cas particulier : les mots en "S", "Z", ou "X" ne changent pas au pluriel.

La dernière lettre du mot est RIGHT$ (MOT$, 1), d'où :

```
7Ø D$ = RIGHT$ (MOT$, 1)
8Ø IF D$ < > "S" AND D$ < > "Z" AND D$ < > "X"
   THEN 11Ø
9Ø PLUR$ = MOT$
1ØØ GOTO 11Ø
```

Listez maintenant l'ensemble du programme pour l'améliorer encore. Sa lecture sera facilitée par l'introduction de lignes blanches pour séparer les différentes parties :

```
10 ' PLURIEL
15 CLS
20 INPUT "QUEL MOT" ; MOT$
25 '
30 IF RIGHT$(MOT$,2)<>"AL" THEN 70
40 L = LEN(MOT$)
50 PLUR$ = LEFT$(MOT$,L-2)+"AUX"
60 GOTO 120
70 D$ = RIGHT$(MOT$,1)
80 IF D$<>"Z" AND D$<>"S" AND D$<>"X"
   THEN 110
90 PLUR$ = MOT$
100 GOTO 120
105 '
110 PLUR$ = MOT$ + "S"
115 '
120 PRINT "PLURIEL : " ; PLUR$
130 PRINT
140 GOTO 20
```

Notez le PRINT seul de la ligne 13Ø. Il a pour effet de faire sauter une ligne.



Chapitre XVI/Des données à la pelle



Vous savez qu'aujourd'hui les météorologues utilisent beaucoup l'ordinateur. D'abord c'est un outil pour tenter de calculer les mouvements des nuages à partir de toutes les données qui interviennent (vitesse du vent, degré hygrométrique, altitude de l'isotherme Ø° ...).
Ceci nécessite de très gros ordinateurs.

Mais des ordinateurs beaucoup plus petits, du même type que le vôtre, sont reliés à des enregistreurs automatiques de températures, des pluviomètres ... Ils font alors des calculs simples : moyennes de températures sur une journée, une semaine, un mois ...

Pour calculer par exemple la température moyenne, relevée à 7 heures le matin, sur une semaine, vous pourriez écrire :

```
10 REM TEMPERATURE MOYENNE
20 T1=18
30 T2=19
40 T3=17.5
50 T4=16
60 T5=20.5
70 T6=18.5
80 T7=19
90 PRINT T1,T2,T3,T4,T5,T6,T7
100 PRINT "MOYENNE:";
110 PRINT (T1+T2+T3+T4+T5+T6+T7)/7
```

Cela va encore. Mais imaginez que vous vouliez la température moyenne annuelle : il y aura 365 lignes différentes à écrire.

Lorsque l'on doit ainsi entrer un grand nombre de données, on a intérêt à utiliser une instruction "étudiée pour" : l'instruction DATA ("données" en anglais) qui est toujours accompagnée de l'instruction READ ("lire").

Voilà le programme équivalent rédigé avec ce couple d'instructions :

```
1Ø REM TEMPERATURE MOYENNE
2Ø DATA 18,19,17.5,16,2Ø.5,18.5,19
3Ø READ T1,T2,T3,T4,T5,T6,T7
4Ø PRINT T1,T2,T3,T4,T5,T6,T7
5Ø PRINT "MOYENNE:";
6Ø PRINT (T1+T2+T3+T4+T5+T6+T7)/7
```

La ligne 2Ø précise la liste des données, séparées chacune par une virgule.
La ligne 3Ø "lit" la suite des données dans l'ordre et
— affecte la 1re donnée (ici 18) à la 1re variable (T1)
— affecte la 2e donnée (ici 19) à la 2e variable (T2)
.... et ainsi de suite
La ligne 4Ø vous permet de vérifier que chaque variable a bien reçu la donnée correspondante et enfin la ligne 5Ø calcule et affiche la valeur de la moyenne des 7 données.

Que se passe-t-il si votre instruction DATA est suivie de moins de données qu'il n'y a de variables dans l'instruction READ ? Effacez la dernière température (19), à la fin de la ligne 1Ø, puis tapez :

```
RUN
?OD Error
```

est la réponse. Elle signifie OUT of DATA, c'est-à-dire "Pas assez de données". En effet à chaque variable doit correspondre une donnée.

Par contre vous pouvez avoir des données en surplus. Par exemple vous pouvez avoir la liste des températures sur un mois et ne lire que celles de la première semaine.
Ajoutez par exemple deux valeurs à la liste des données :

```
1Ø DATA 18,19,17.5,16,2Ø.5,18.5,19,2Ø,21
RUN
```

Le résultat est identique : en ligne 2Ø on demande seulement la lecture des 7 premières données.

Changez maintenant le numéro de la ligne qui contient l'instruction DATA : remplacez le 2Ø par 99, et n'oubliez pas d'effacer la ligne 2Ø (en tapant simplement 2Ø puis [ENTREE]).

```
10 REM TEMPERATURE MOYENNE
30 READ T1,T2,T3,T4,T5,T6,T7
40 PRINT T1,T2,T3,T4,T5,T6,T7
50 PRINT "MOYENNE:";
60 PRINT (T1+T2+T3+T4+T5+T6+T7)/7
99 DATA 18,19,17.5,16,20.5,18.5,19
```

Le programme s'exécute de la même manière.
La liste de données peut donc se placer n'importe où dans le programme (même après l'instruction END).

Si le programme est important on place souvent ces données en dernière ligne pour y accéder facilement lorsque l'on veut les modifier.

Simplifier la lecture des données avec une boucle FOR ... NEXT

Si vous regardez les lignes 2Ø et 3Ø du programme précédent, elles répètent chacune 7 fois la même opération : lire une donnée (L.2Ø) et l'imprimer (L.3Ø).

Lorsque l'ordinateur doit répéter ainsi la même action, il est plus simple de placer cette action dans une boucle :

```
10 REM TEMPERATURE MOYENNE
20 FOR I=1 TO 7
30    READ T
40    PRINT T,
50 NEXT I
99 DATA 18,19,17.5,16,20.5,18.5,19
```

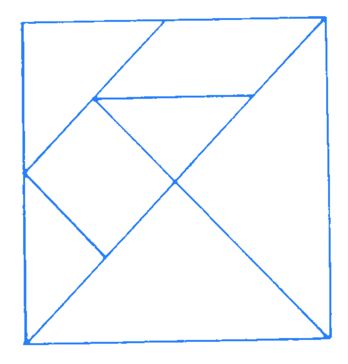
Au premier tour de la boucle, l'instruction READ T lit la première donnée et l'imprime ; au deuxième tour elle va automatiquement chercher la deuxième donnée car la première a déjà été utilisée ... et ainsi de suite.
Voyez-vous l'intérêt de cette boucle de lecture si vous aviez à calculer la moyenne annuelle ?

Mais il manque encore dans ce programme le calcul de la moyenne. Il suffit de calculer à chaque tour la somme des températures déjà lues :
— au 1e tour cette somme sera ST = T1
— au 2e tour cette somme sera ST = T1 + T2
— au 3e tour cette somme sera ST = T1 + T2 + T3
.....
A chaque tour il faut donc ajouter à la valeur précédente de ST la nouvelle température lue.
Précisons encore qu'au début du programme la somme ST est nulle :

```
10 REM TEMPERATURE MOYENNE
15 ST=0
20 FOR I=1 TO 7
30    READ T
40    PRINT T,
45    ST=ST+T
50 NEXT I
60 PRINT "MOYENNE:";ST/7
99 DATA 18,19,17.5,16,20.5,18.5,19
```

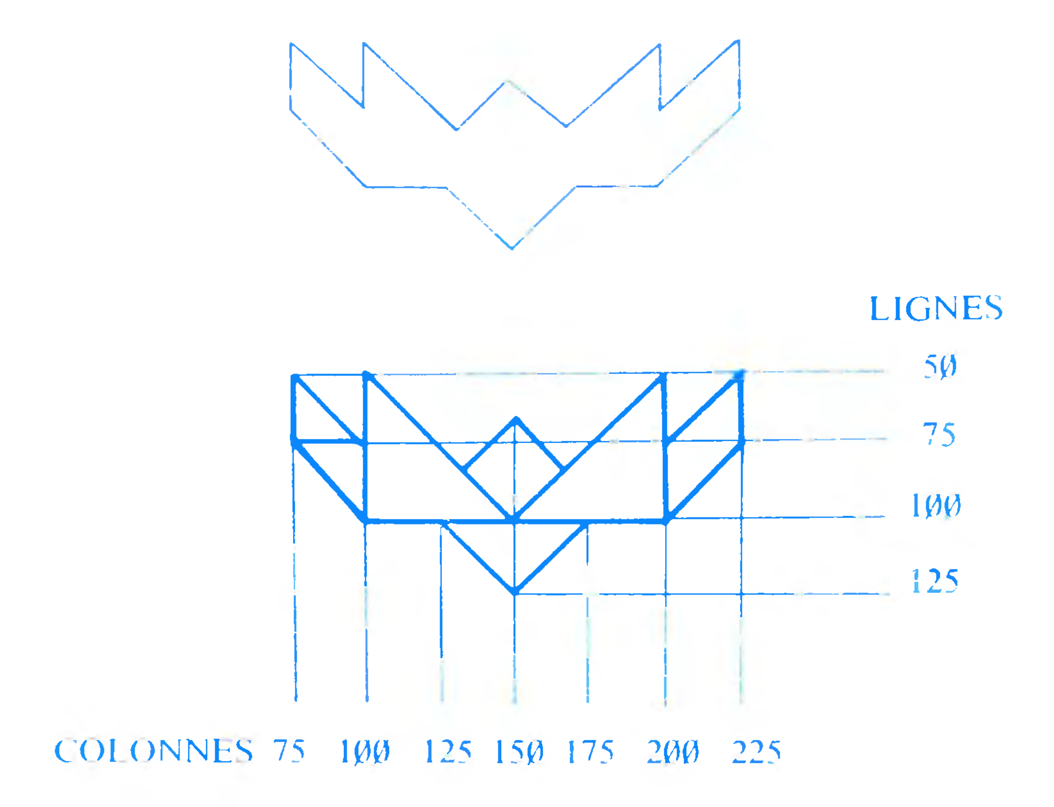
Le Tangram

Connaissez-vous le Tangram ?
Ce n'est ni une montagne russe, ni un gâteau tunisien, mais un puzzle chinois. Les morceaux du puzzle peuvent se regrouper sur un carré :



Mais avec ces mêmes morceaux on peut former un très grand nombre de dessins, figuratifs ou abstraits.
Ce jeu ne nécessite qu'une feuille de papier et une paire de ciseaux.

Nous allons dessiner à l'écran une figure formée à l'aide de ce puzzle :



Vous pourrez imaginer à votre gré un oiseau maléfique ou un véhicule extra-terrestre en piqué sur notre planète.
Le tracé du dessin nécessite la donnée des positions de chacun des points (colonne et ligne).
Il y a 16 points différents donc il sera intéressant de ranger ces données dans une instruction DATA .

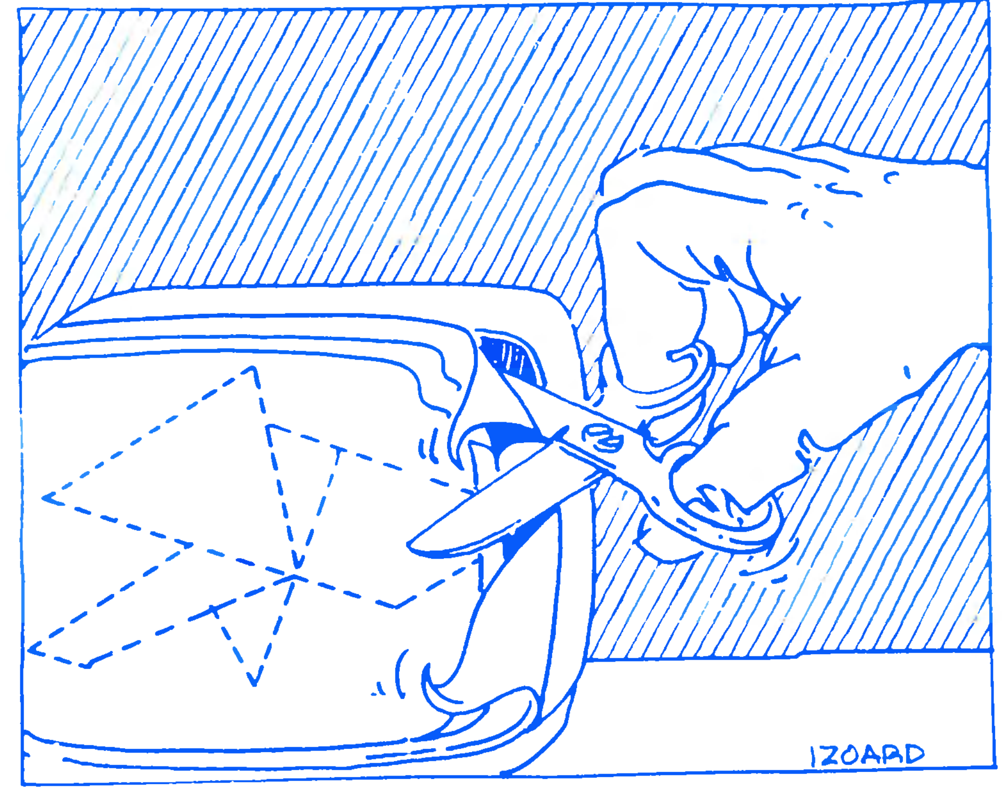
Sur une même ligne DATA , vous pouvez ranger des données de significations différentes. Ici nous placerons les unes à la suite des autres les coordonnées des points : un numéro de colonne, un numéro de ligne, un numéro de colonne, un numéro de ligne ...

NOTE

Si vous voulez calculer les numéros de colonnes et lignes qui manquent sur le schéma, nous vous rappelons ce qu'a dit Pythagore il y a environ 26 siècles : si la longueur du côté d'un carré est "a", alors l'hypothénuse "b" est telle que : $a^2 + a^2 = b^2$ d'où $b = a\sqrt{2}$

```
10 ' OISEAU
20 CLS
30 READ C,L
40 PSET (C,L)
50 FOR I=1 TO 16
60    READ C,L
70    LINE -(C,L)
80 NEXT I
100 DATA 100,100,   75, 75,    75 , 50,
         100, 75,  100, 50,   132, 82,
         150, 65,  168, 82,   200, 50,
         200, 75,  225, 50,   225, 75,
         200,100,  175,100,   150,125,
         125,100,  100,100
```

Les lignes 3Ø et 4Ø allument le premier point de la liste des données. A chaque tour de boucle il y a lecture des 2 données suivantes C et L.
Il faut donc 16 tours pour tracer les 16 traits reliant tous les points entre eux.
Comme vous pouvez le constater en ligne 1ØØ du programme, il est possible de remplacer un "blanc" par plusieurs "blancs" dans une instruction. Ceci permet de mieux séparer les différents éléments et facilite la lecture de l'instruction comme ici.



Vous pouvez maintenant essayer d'enrichir le programme en lui faisant tracer les traits qui délimitent les différentes pièces du puzzle. Vous pouvez aussi, avec ce même programme faire le dessin de votre choix : il suffit de changer les coordonnées des points dans la ligne DATA et remplacer le nombre 16 par le nombre de points à relier.

Votre travail sera facilité par l'utilisation d'une grille comportant les numéros de colonnes et de lignes, de 5 en 5 ou de 1Ø en 1Ø suivant la précision de votre dessin.

Un fichier téléphonique sur ordinateur

Dans les deux exemples précédents, les données qui étaient entrées dans la mémoire de l'ordinateur par l'instruction DATA étaient des nombres.
Il est possible de rentrer de la même manière des chaînes de caractères.
Regardez le programme suivant :

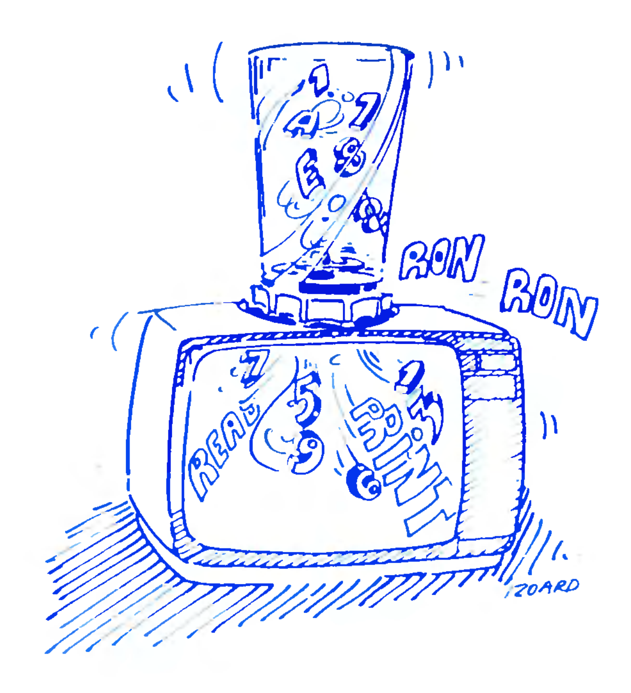
```
10 READ N$,P$,T$
20 PRINT N$,P$,T$
30 DATA "DURAND", "PIERRE", "666-66-66"
```

Pour faciliter votre tache dans l'introduction des données "chaînes de caractères", les guillemets sont facultatifs dans l'instruction DATA .
C'est dans l'instruction READ que le type de chaque donnée est précisé : si le nom de variable se termine par $, la donnée correspondante doit être une chaîne de caractères ; si le nom ne se termine pas par $, la donnée doit être un nombre.
Vous pouvez donc écrire de manière équivalente :

3Ø DATA DURAND, PIERRE, 666 – 66 – 66

Vous pouvez même mélanger dans une instruction DATA des données numériques et des données "chaînes de caractères" :

```
10 READ A$,N$,T$
20 PRINT A$,N$,T$
30 DATA 1,DURAND,666-66-66
```



Dans ce cas le numéro de téléphone est bien considéré comme une chaîne de caractères puisque son nom est T$.

Voulez-vous mettre votre fichier téléphonique en mémoire ? Naturellement on n'a pas toujours son ordinateur sous la main (même si c'est un micro) lorsqu'on cherche un numéro de téléphone.
Mais par contre si vous avez des centaines de numéros à stocker ou si vous avez parfois des difficultés à vous relire, ce petit annuaire électronique pourra vous être utile.

Que devra faire le programme ?
— demander le nom désiré
— fournir le numéro de téléphone correspondant
— conserver en données la liste des noms avec leurs numéros

Lorsque l'utilisateur donne le nom qui l'intéresse, il faut que le programme compare ce nom avec les noms successifs de sa liste jusqu'à ce qu'il trouve le même :

```
10 ' ANNUAIRE
20 CLS
30 INPUT "QUEL NOM VOULEZ-VOUS" ;NOM$
40 READ N$,TEL$
50 IF N$<>NOM$ THEN 40
60 PRINT N$,TEL$
100 DATA DUPONT , 12 34 56
110 DATA DURAND , 23 45 67
120 DATA MARTIN , 34 56 78
```

Pour mieux séparer les données, nous avons placé chaque nom sur une nouvelle ligne DATA. On peut ainsi répartir les données sur des lignes différentes : elles seront toujours lues dans l'ordre.

L'instruction READ lit les deux premières données, ici DUPONT et 12 34 56 (ligne 4Ø).

Si DUPONT n'est pas le nom cherché N$, alors on retourne en ligne 4Ø pour lire deux nouvelles données.
Et ainsi de suite jusqu'à ce que le dépouillement de toutes les données deux par deux aboutisse au nom cherché. Alors l'exécution se poursuit en ligne 6Ø par l'impression du nom et du numéro.

Testez votre programme.

Que se passe-t-il si vous demandez un nom qui n'est pas dans la liste (ou que vous orthographiez mal un nom)?
L'ordinateur lit toute la liste sans trouver le nom cherché et cherche donc encore une donnée après le dernier nom ce qui provoque le message d'erreur :
?OD Error

Le programme donné en exemple à la fin du livre évite l'apparition de ce message et affiche dans ce cas : "X" n'est pas dans l'annuaire.

Chapitre XVII/Des tableaux

Faites votre déclaration

Dans le chapitre précédent nous avons vu comment ranger des données avec une instruction DATA . Ceci est très pratique lorsque l'on utilise toutes les données en même temps.
Mais si l'on s'intéresse d'abord à la 1Øe, puis à la 5e, qu'ensuite on veuille modifier la 25e... Cela ne sera plus aussi pratique.
En effet à chaque fois il faut relire la liste de toutes les données jusqu'à celle qui nous intéresse.

Dans ce cas-là, on range les données dans un "tableau", c'est-à-dire en leur donnant un numéro d'ordre.
Tous les éléments d'un tableau porteront le même nom, le nom du tableau, suivi entre parenthèses de leur numéro d'ordre ou "indice" :



Vous pouvez utiliser l'indice Ø et alors ce tableau comportera 21 éléments. Mais en général il est plus simple de commencer à l'indice 1. C'est ce que nous ferons.

Pour ranger tous les éléments du tableau, l'ordinateur doit réserver assez de place en mémoire. Pour cela, il faut préciser avant d'utiliser un tableau le nombre maximum d'éléments que celui-ci pourra contenir.
C'est l'instruction DIM qui réserve cette place nécessaire :

```
DIM T(1ØØ)
```

cette instruction "déclare" un tableau de nom T comprenant 1ØØ éléments (en fait 1Ø1 si on utilise l'indice Ø).

Un tableau étant un ensemble de variables, son nom doit suivre les mêmes règles que les noms de variables :

```
DIM PRIX (15)
DIM N2 (52)
DIM A (4ØØ)
```

sont des déclarations de tableaux valables : la réponse est OK.

Une variable numérique et un tableau peuvent avoir le même nom : par exemple A et A(5ØØ), l'ordinateur fera la différence. Il ne confondra pas non plus avec une chaîne de caractères appelée A$.

Tout a des limites



Tapez maintenant NEW pour vider la mémoire de travail puis déclarez le tableau suivant :

DIM T(1ØØØØ)

La réponse est :

?OM Error

Ce message (Out of Memory) signifie que vous dépassez la capacité de la mémoire de l'ordinateur : elle ne peut pas contenir un tableau de 1ØØØØ éléments.

Suivant la capacité de votre machine (8 ou 24 K), vous pouvez utiliser des tableaux d'environ 18ØØ ou 5ØØØ éléments. C'est déjà pas mal (mais il restera alors peu de place en mémoire pour écrire le programme).

Si vous voulez vérifier à un moment donné quelle est la place qui reste libre dans la mémoire, il suffit de taper :

?FRE (Ø)

(''free'' = libre)

La réponse fournie indique le nombre d'octets libres dans l'espace qui vous est réservé pour travailler dans la mémoire. Dès que ce nombre avoisine Ø, il est temps de s'arrêter et de prendre des mesures d'urgence : supprimer quelques lignes inutiles par une meilleure programmation ou courir acheter de la mémoire supplémentaire.

Remplir les cases

Lorsque vous avez réservé la place nécessaire en mémoire pour le tableau, vous pouvez affecter une valeur à chaque élément du tableau :

```
NEW
10 DIM T(10)
20 T(1)=10
30 T(2)=20
40 T(3)=30
```

Vous n'êtes pas obligé de remplir votre tableau en une seule fois : on peut garder des cases "en réserve" (elles ont la valeur Ø).

Faites RUN.

Lorsque des variables du tableau ont reçu des valeurs numériques, elles sont utilisées comme n'importe quelle variable :

```
?T(2)*T(3)
?T(1)+1ØØ
```

Mais un tableau peut être rempli encore plus facilement avec l'instruction INPUT . Déclarons un nouveau tableau A :

```
NEW
10 CLS
20 DIM A(12)
30 FOR I=1 TO 12
40    PRINT "ELEMENT NUMERO";I;
50    INPUT A(I)
60 NEXT I
RUN
```

Donnez à chaque élément la valeur de votre choix. Remarquez seulement le point-virgule à la fin de la ligne 4Ø, il implique l'affichage du point d'interrogation de l'instruction INPUT après le numéro I de l'élément.

A quoi sert un tableau ?

Souvent on a seulement besoin d'aller chercher dans le tableau la valeur de tel ou tel élément : prix de l'article n° 45, température du jour n° 12, score du joueur n° 3...
Mais on peut aussi avoir besoin du tableau pour des opérations plus complexes : faire la moyenne des éléments, chercher quel est le plus grand, le plus petit, les classer par ordre croissant...

Cherchons par exemple comment trouver l'élément le plus grand

du tableau A que vous avez rempli dans le programme précédent. Appelons MAX cet élément cherché (et inconnu !).
Regardons les éléments les uns après les autres :
— le n° 1 : s'il était tout seul, il serait le plus grand (M. de la Palice n'aurait pas dit mieux). Prenons donc MAX = A(1) pour commencer.
— le n° 2 : s'il est supérieur à MAX alors MAX = A(2), sinon nous gardons l'ancien MAX.
— le n°3 : s'il est supérieur à MAX alors MAX = A(3), sinon nous gardons l'ancien MAX.
... et ainsi de suite.

Rajoutons maintenant au programme précédent la recherche de l'élément le plus grand :

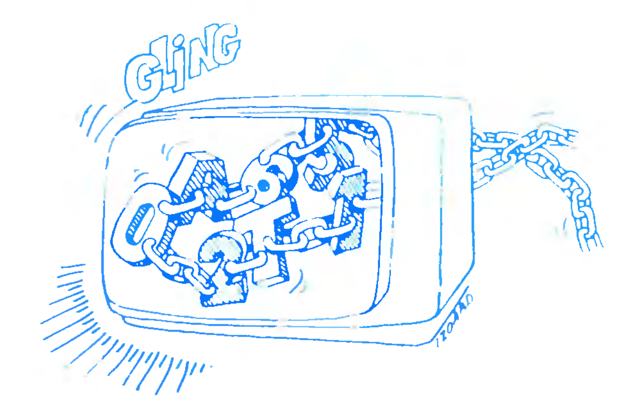
```
70 MAX=A(1)
80 FOR I=2 TO 12
90    IF A(I)>MAX THEN MAX=A(I)
100 NEXT I
110 PRINT "MAXIMUM ",MAX
```

Testez le programme : vous obtenez l'élément maximum du tableau.

Avec la même méthode, il n'est pas difficile de trouver le minimum ; il suffit d'utiliser à la place de MAX une variable MIN et de tester en ligne 90 si A(I) est inférieur à MIN.

Tableaux de chaînes

Il ne s'agit pas d'œuvres d'art moderne représentant des chaînes, mais de suites (ou tableaux) de chaînes de caractères.
De même qu'il existe deux types de variables : les variables numériques et les variables "chaînes de caractères", il existe deux types de tableaux correspondants.



Le nom d'un tableau chaîne de caractères doit se terminer par $ et il faut déclarer sa taille, c'est-à-dire le nombre d'éléments avant de l'utiliser.

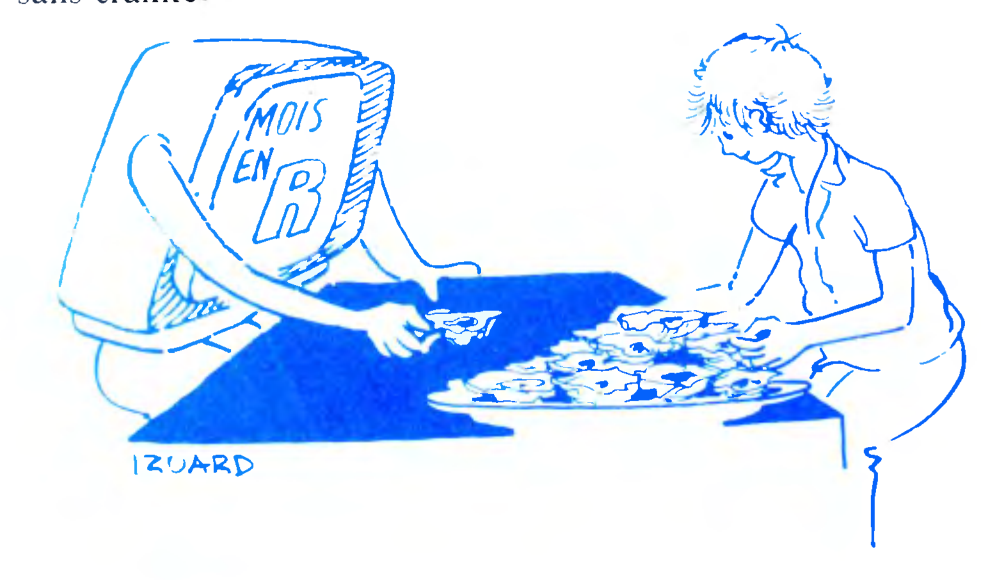
Voici par exemple un tableau à douze éléments qui contient les noms des douze mois de l'année :

```
10 ' MOIS
20 DIM M$(12)
30 M$(1)="JANVIER"
40 M$(2)="FEVRIER"
50 M$(3)="MARS"
60 M$(4)="AVRIL"
70 M$(5)="MAI"
80 M$(6)="JUIN"
90 M$(7)="JUILLET"
100 M$(8)="AOUT"
110 M$(9)="SEPTEMBRE"
120 M$(10)="OCTOBRE"
130 M$(11)="NOVEMBRE"
140 M$(12)="DECEMBRE"
```

Avec ce tableau, vous pouvez facilement convertir une date donnée sous la forme "15-1Ø-1982" en "15 OCTOBRE 1982" :

```
145 '
150 INPUT "JOUR,MOIS,ANNEE";J,M,A
160 PRINT J;M$(M);A
```

Vous pourrez aussi chercher quels sont les mois en "R" (mois contenant la lettre R) qui, avant l'ère de la congélation, étaient les seuls mois pendant lesquels on pouvait manger des huîtres sans crainte.



Pour savoir si la lettre R figure dans un mot, on peut tester toutes les lettres du mot successivement avec la fonction MID$. Mais il existe une fonction qui fait cela automatiquement : la fonction INSTR .

Pour voir comment fonctionne cette instruction, testez les lignes suivantes :

```
?INSTR ("JANVIER", "R")
?INSTR ("JANVIER", "A")
?INSTR ("JANVIER", "Z")
```

Si la lettre figure dans le mot, la fonction INSTR donne sa position dans le mot et si la lettre n'y figure pas, la réponse est Ø.

Le tri des mois en "R" se fera donc de la manière suivante :

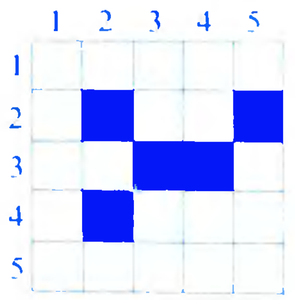
```
170 '
180 FOR M=1 TO 12
190  IF INSTR(M$(M),"R")=0 THEN 210
200  PRINT M$(M);"BON POUR LES HUITRES"
210 NEXT M
```

Si vous voulez faire de cette partie de programme un programme distinct du précédent, il vous faudra remplir à nouveau le tableau des mois. En effet à chaque exécution d'un programme, c'est-à-dire à chaque fois que l'on fait RUN , toutes les variables et tous les tableaux sont remis à zéro.

Un tableau à deux dimensions

Si vous voulez imaginer un jeu de bataille navale, ou tout autre jeu qui utilise une grille, vous devrez préciser l'état de chaque case de la grille : vide ou bien pleine (par exemple par un chiffre Ø ou 1).

Sur la petite grille suivante, chaque case est répérée par son numéro de colonne (C) et de ligne (L), donc par le couple de chiffres (C, L).

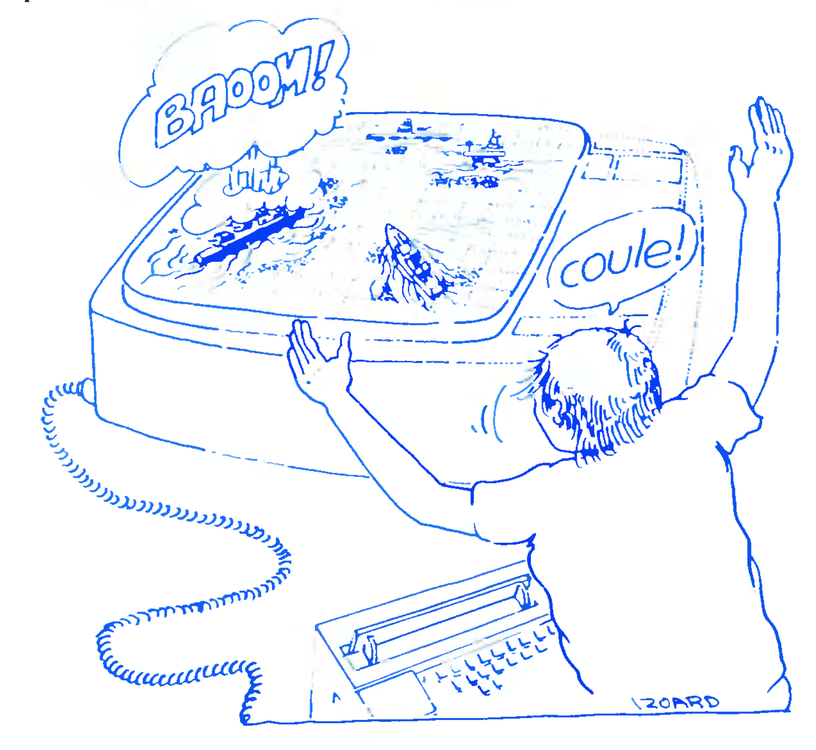


Ainsi les cases pleines sont les cases repérées par les couples de chiffres suivants :
(2,2)
(2,4)
(3,3)
(4,3)
(5,2)
Toutes les autres cases sont vides.

On peut ranger ces résultats en mémoire en créant un tableau de variables. La valeur de la variable sera Ø ou 1 suivant le cas et cette variable dépendra de deux indices : C et L.
On peut imaginer ces valeurs rangées dans un vrai "tableau" (c'est de là que vient ce nom) :

T(1,1)	T(2,1)	T(3,1)	T(4,1)	T(5,1)
T(1,2)	T(2,2)	T(3,2)	T(4,2)	T(5,2)
T(1,3)	T(2,3)	T(3,3)	T(4,3)	T(5,3)
T(1,4)	T(2,4)	T(3,4)	T(4,4)	T(5,4)
T(1,5)	T(2,5)	T(3,5)	T(4,5)	T(5,5)

De même que pour les tableaux à une seule dimension, il faut au préalable déclarer la taille du tableau, c'est-à-dire la valeur maximale de chaque indice.
Ici chaque indice va jusqu'à 5. La déclaration du tableau se fera donc par l'instruction DIM T (5,5) .



Remplissons le tableau en précisant les éléments qui sont égaux à 1 (tous les autres seront nuls) :

```
NEW
10 ' BATAILLE NAVALE
20 DIM T(5,5)
30 T(2,2)=1 : T(2,4)=1 : T(3,3)=1
40 T(4,3)=1 : T(5,2)=1
```

Maintenant que le bateau est placé, vous pouvez demander au joueur quelle case il veut tester :

```
50 '
60 PRINT "QUELLE CASE VOULEZ-VOUS ?"
70 INPUT "COLONNE";C
80 INPUT "LIGNE";L
```

Puis lui répondre s'il a touché le bateau ou non :

```
90 '
100 IF T(C,L)=1 THEN 120
110 PRINT "RIEN" : GOTO 60
120 PRINT "TOUCHE" : GOTO 60
```

Listez le programme et testez-le.

On peut encore perfectionner le programme : lorsque les 5 cases du bateau ont été touchées, la réponse devrait être "coulé".

Rajoutons donc un compteur de coups gagnants, c'est-à-dire une variable G qui augmente d'une unité à chaque case touchée. En même temps, ajoutons un compteur de coups qui dira au joueur en combien de coups il a gagné :

```
10 ' BATAILLE NAVALE
20 DIM T(5,5)
30 T(2,2)=1 : T(2,4)=1 : T(3,3)=1
40 T(4,3)=1 : T(5,2)=1
50 '
60 PRINT "QUELLE CASE VOULEZ-VOUS ?"
70 INPUT "COLONNE";C
80 INPUT "LIGNE";L
90 W=W+1 ' COMPTEUR DE COUPS
100 IF T(C,L)=1 THEN 120
110 PRINT "RIEN" : GOTO 60
120 G=G+1 ' COMPTEUR DE COUPS GAGNANTS
130 IF G=5 THEN 150
140 PRINT "TOUCHE" : GOTO 60
150 PRINT "COULE"
160 PRINT
170 PRINT "VOUS AVEZ GAGNE EN";W;"COUPS"
```

Vous pouvez encore enrichir le programme en prenant une grille plus grande avec plusieurs bateaux.

Exemples

L'essentiel de la programmation en BASIC a maintenant été vu : variables numériques, répétitions avec boucle, branchement conditionnel avec IF ... THEN , introduction de données par INPUT ou DATA , décomposition en sous-programmes, variables chaînes de caractères, tableaux...

Tous ces outils du programmeur vont être mis en œuvre dans les exemples qui suivent. Ces exemples résolvent des problèmes plus complexes que ceux que nous avons rencontrés chapitre après chapitre.
Ils sont là pour ouvrir des perspectives sur ce que vous pourrez faire vous-même.

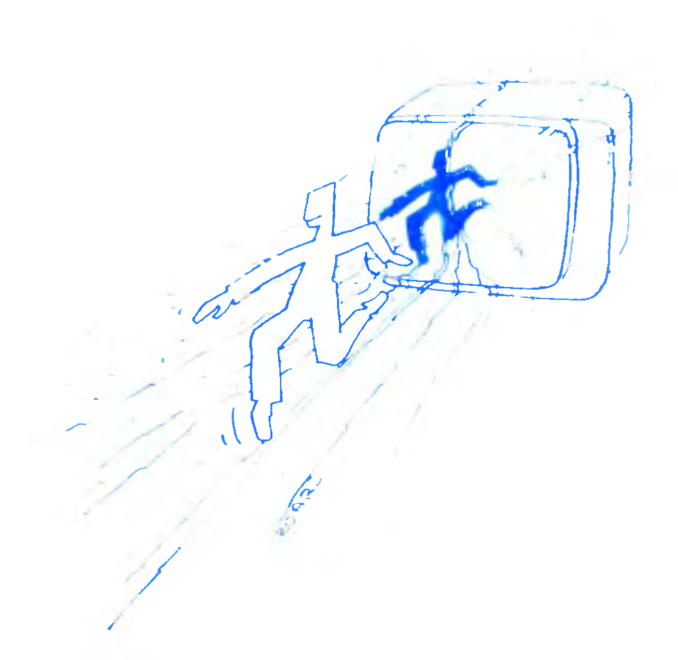
Dans chaque cas le problème est posé et vous pourrez essayer de le résoudre, quitte à le simplifier, avant de regarder notre solution. Cette solution n'est d'ailleurs qu'une solution parmi d'autres possibles (et peut-être meilleures).

Plusieurs solutions utilisent des instructions du BASIC TO7 qui n'ont pas été abordées dans ce livre.
Dans certains cas, c'est pour faciliter l'écriture du programme ou la présentation à l'écran, on aurait pu s'en passer mais il était dommage de ne pas en profiter.
Dans tous les cas, elles sont expliquées dans le commentaire de chaque exemple. Vous trouverez tous les renseignements complémentaires relatifs à ces instructions dans le Manuel de référence Basic.

Exemple 1/Bonhomme animé

But du programme

Dessiner un bonhomme qui lève et baisse les bras alternativement. On se contentera dans un premier temps de représenter sa tête par un carré et le tronc et les membres par des droites.



Commentaire du programme

Le programme comporte trois parties essentielles :

— Le dessin du bonhomme sans ses bras (lignes 2Ø-6Ø) ;

— L'animation des bras, qui constitue la partie "principale" du programme ;

— Les deux sous-programmes qui dessinent les bras levés et les bras baissés (lignes 1ØØØ-1Ø2Ø et 2ØØØ-2Ø2Ø).

Le procédé utilisé pour l'animation est simple. Il consiste à dessiner puis à effacer successivement les bras levés ou baissés : pour effacer une portion de dessin, il suffit de le tracer dans la couleur du fond.

Il faut donc dès le départ bien choisir dans quelles couleurs on travaille. C'est le rôle de l'instruction SCREEN en ligne 3Ø : on dessine en noir (Ø) sur fond bleu clair (6).

Entre le tracé et l'effacement, il faut attendre un minimum de temps, sinon le mouvement est tellement rapide que l'on ne voit pas grand-chose. Il y a donc deux boucles de temporisation en lignes 9Ø et 12Ø.

Les bras levés sont tracés en ligne 8Ø puis effacés en ligne 1ØØ, puis les bras baissés sont tracés en ligne 11Ø et effacés en ligne 13Ø. Et enfin, la boucle étant infinie, il faudra l'arrêter par CNT C.

Le dessin des bras a été placé dans deux sous-programmes : le premier pour les bras levés de 1ØØØ à 1Ø2Ø, le deuxième pour les bras baissés de 2ØØØ à 2Ø2Ø.
La seule variable qui change à chaque exécution du sous-programme est la couleur C qui passe alternativement de Ø à 6.

Il est possible maintenant de poursuivre l'idée en donnant d'autres positions aux bras (par exemple vers le bas), on aura alors une meilleure impression de continuité du mouvement. Mais vous pouvez aussi faire bouger les jambes, la tête, le tronc et régler la vitesse du mouvement en agissant sur la durée des boucles de temporisation.

```
10 ' BONHOMME ANIME
20 CLS : SCREEN 0,6,6
30 BOX (95,100)-(105,110)
40 LINE (100,110)-(100,140)
50 LINE (100,140)-(80,170)
60 LINE (100,140)-(120,170)
70 '
80  C=0 : GOSUB 1000
90  FOR I=1 TO 100 : NEXT I
100 C=6 : GOSUB 1000
110 C=0 : GOSUB 2000
120 FOR I=1 TO 100 : NEXT I
130 C=6 : GOSUB 2000
140 GOTO 80
999 '
1000 LINE (100,120)-(70,100),C
1010 LINE (100,120)-(130,100),C
1020 RETURN
1999 '
2000 LINE (100,120)-(70,120),C
2010 LINE (100,120)-(130,120),C
2020 RETURN
```

Exemple 2/Bande annonce

But du programme

Afficher un message au milieu de l'écran en le faisant défiler continuellement comme les annonces lumineuses de certains magasins.
Le message est toujours plus long que la bande sur laquelle il s'affiche. Il défile en boucle, le début étant séparé de la fin par quelques points.



Commentaire du programme

La première partie du programme est consacrée à fixer le cadre initial : effacer l'écran et choisir les couleurs standard (7, Ø, Ø), mettre la phrase à écrire dans une variable (PH$), la faire précéder d'une vingtaine de points (ligne 5Ø), déterminer sa longueur et la conserver dans la variable L (ceci nous servira par la suite) et enfin choisir dans quelles couleurs faire apparaître le texte.
L'instruction :

```
COLOR 1, 6
```

précise que les caractères seront de couleur 1 et que les rectangles dans lesquels s'inscrivent chaque caractères seront de couleur 6.

Le défilement du texte à l'écran au milieu de l'écran va se faire de la manière suivante :

1° Positionner le curseur en colonne 1Ø, ligne 1Ø ;
2° Ecrire les 2Ø premiers caractères de la phrase ;
3° Fabriquer une nouvelle phrase en décalant tous les caractères d'une case à gauche, c'est-à-dire en retirant le premier pour le mettre à la fin ;
4° Reprendre en 1°.

```
10 'BANDE ANNONCE
20 CLS : SCREEN 7,0,0
30 PH$="LE BASIC DU TO 7 EST A LA FOIS"
40 PH$=PH$+"SIMPLE ET TRES RICHE"
50 PH$=".....................  "+PH$
60 L=LEN(PH$)
70 COLOR 1,6
80 ' DEFILEMENT
90 LOCATE 10,10
100 PRINT LEFT$(PH$,20)
110 FOR W=1 TO 100 : NEXT W
120 PH$=RIGHT$(PH$,L-1)+LEFT$(PH$,1)
130 GOTO 80
```

Amélioration possible

Le programme tel qu'il est ne s'arrête que par CNT C. Or il faut deux doigts pour appuyer sur CNT et C, c'est bien fatiguant.
Il existe une fonction qui permet de savoir si on a appuyé sur une touche du clavier et dans ce cas laquelle. Ici nous pouvons l'utiliser pour arrêter le programme en tapant sur n'importe quelle touche (voir exemple 11).
Cette fonction s'écrit INKEY$

Remplacez la ligne 13Ø par :

```
130 R$=INKEY$
140 IF R$="" THEN 80
150 COLOR 7,0
```

Si aucune touche n'a été touchée, c'est-à-dire si la variable R$ est vide alors on recommence en ligne 8Ø. Dans le cas contraire, le programme continue en séquence, donc s'arrête puisqu'il n'y a plus rien.
Vous pouvez retrouver la touche utilisée par :

```
?R$
```

Exemple 3/Annuaire électronique

But du programme

Remplir la même fonction que l'annuaire édité sur papier, mais pour un nombre de personnes plus limité.
Connaissant le nom de la personne que vous voulez appeler, le programme vous donne son prénom, son adresse et son numéro de téléphone.
Si plusieurs personnes répondent à ce nom, elles vous sont toutes indiquées. A l'opposé, si personne ne correspond au nom que vous donnez, le programme vous l'indique également.

Commentaire

La liste des noms, prénoms, adresses et numéros de téléphone est entrée avec l'instruction DATA à la fin du programme à partir de la ligne 1ØØØ.
Pour la clarté de l'écriture et pour pouvoir introduire de nouveaux noms et en supprimer d'autres, il y a une ligne DATA par personne avec dans l'ordre : le nom, le prénom, l'adresse et le numéro de téléphone.
Remarquez que l'adresse, qui constitue une seule chaîne de caractères, est écrite entre guillemets car elle peut contenir une virgule.

Notez bien que chaque ligne DATA doit contenir ces quatre informations sinon la lecture des données suivantes se trouve décalée et vous ne retrouvez plus aucun nom.

Après initialisation et affichage du titre (il est toujours utile de mettre un titre à un programme), on vous demande quel nom vous cherchez.
Le nom fourni est stocké dans la variable NOM$.
Ensuite les lignes 7Ø à 13Ø constituent la boucle de recherche du nom dans la liste des DATA :
1° Lire un nom dans la liste, avec les informations qui le suivent ;
2° Si c'est le dernier de la liste, sortir de la boucle ;
3° Si le nom lu n'est pas identique à celui fourni dans NOM$, aller à la fin de boucle. Sinon marquer le fait qu'on a trouvé un nom (en posant T = 1) et afficher le nom et les renseignements : prénom, adresse, téléphone ;
4° Fin de boucle : aller en 1°.

Les dernières données de la liste sont un nom, prénom ... fictifs.

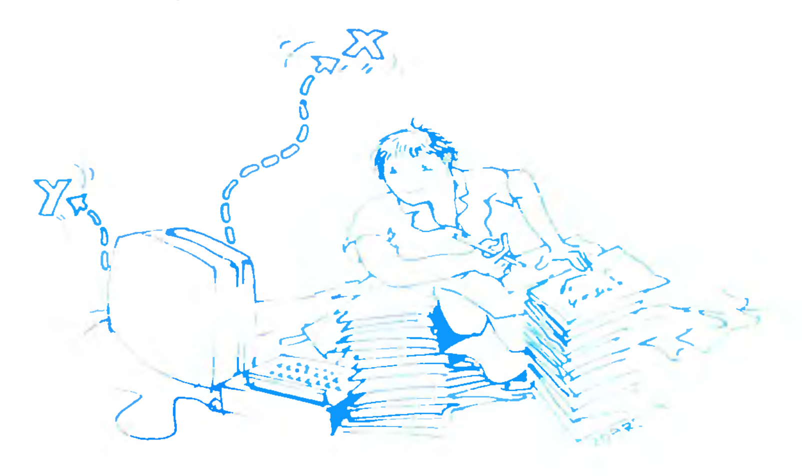
Ce procédé évite l'apparition d'un message d'erreur ?OD Error (manque de données) lorsqu'on demande un nom qui n'est pas dans la liste.
La variable T sert à marquer le fait qu'on a rencontré au moins une fois dans la liste le nom cherché.
A la sortie de la boucle, si aucun nom identique n'a été rencontré, c'est-à-dire si T = Ø, le programme indique que le nom n'est pas dans l'annuaire.

```
10 ' ANNUAIRE
20 CLS
30 LOCATE 6,6
40 PRINT "CONSULTATION DE L'ANNUAIRE"
50 PRINT
60 INPUT "QUEL NOM VOULEZ-VOUS ";NOM$
70 T=0
80 READ N$,P$,ADR$,TEL$
90 IF N$="XXX" THEN 150
100 IF N$<>NOM$ THEN 140
110 T=1
120 PRINT
130 PRINTNOM$,P$,ADR$,TEL$
140 GOTO 80
150 ' FIN DE LA LISTE
160 IF T>0 THEN 180
170 PRINT NOM$;
    " N'EST PAS DANS L'ANNUAIRE"
180 END
999 '
1000 ' LISTE DES NOMS
1010 DATA DUPONT , PIERRE ,
     "8,ALLEE DES ROSIERS CARPENTRAS",
      12 34 56
1020 DATA DURAND , JACQUES,
     "2,RUE DU PONT EN BOIS MONTARGIS",
     23 45 67
1030 DATA MARTIN ,LOUIS,
     "57,GRANDE RUE STRASBOURG",
     37 32 10
1040 DATA MARTIN ,SYLVESTRE,
     "13,RUE DU MOULIN FONTAINEBLEAU",
     43 73 35
1050 DATA LEPETIT,ALBERT,
     "ROUTE DES CHAMPS CAMENBERT",
     25 73 85
9999 DATA XXX,XXX,XXX,XXX
```

Exemple 4/Sinus

But du programme

Dessiner la courbe $y = \sin x$ avec ses axes.



Commentaire du programme

— Tracé des axes :

Sur la première ligne à l'écran on a affiché le nom de la courbe : "SINUS X".
L'axe vertical est sur la colonne n° 19 (ligne 6Ø).
L'axe horizontal est sur la ligne n° 99 (ligne 8Ø).
L'origine du repère est donc en (19, 99).
On a ajouté des flèches aux extrémités des axes avec le symbole ^ (exponentiation) pour l'axe vertical et le symbole > (supérieur à) pour l'axe horizontal (lignes 5Ø et 7Ø du programme). Les axes ont d'ailleurs été placés pour tomber au milieu de ces caractères ^ et > .

L'affichage du symbole > est fait par un nouvel usage de l'instruction PSET :

```
PSET (2, 1) ">",1
```

Cette instruction est équivalente à la suite d'instruction suivante :

```
LOCATE (2, 1)
COLOR 1
PRINT ">"
```

Attention : dans cet usage de l'instruction PSET , c'est-à-dire lorsqu'elle est suivie d'un caractère entre guillemets, les numéros

de colonne et de ligne sont relatifs à des lignes et des colonnes de caractères. (Le n° de ligne est compris entre Ø et 24 ; le n° de colonne est compris entre Ø et 39).

— Tracé de la courbe :

Lorsqu'on veut tracer une courbe à l'écran, il ne faut pas oublier que les numéros de lignes à l'écran augmentent vers le bas alors qu'en général on oriente l'axe vertical vers le haut.

On a choisi ici une échelle différente pour chaque axe :
7Ø sur l'axe vertical
2ØØ sur l'axe horizontal
Les numéros de colonne (COL) et de ligne (LI) du point M sont donc :
COL = 19 + 2ØØ*X
LI = 99 – 7Ø*Y
Vous pouvez modifier ces valeurs 2ØØ et 7Ø pour voir leur rôle.
Le point de départ de la courbe est l'origine (COL = 19, LI = 99).
La boucle (1ØØ-14Ø) joint chaque point au point suivant, la distance entre 2 points est fixée par le pas de variation de X (Ø.Ø1).
Enfin la dernière ligne du programme positionne la réponse "OK" en bas de l'écran.

```
10 '  SINUS
20 CLS
30 SCREEN 7,0,0
40 PRINT "SINUS X"
50 PSET (2,1)"^",1
60 LINE (19,10)-(19,190)
70 PSET (39,12)">",1
80 LINE (10,99)-(315,99)
90 PI=3.1416
100 FOR X=0 TO 1.4 STEP 0.01
110  Y=SIN(2*PI*X)
120  COL=200*X+19 : LI=-70*Y+99
130  LINE -(COL,LI),1
140 NEXT X
150 LOCATE 1,22 : COLOR 7
```

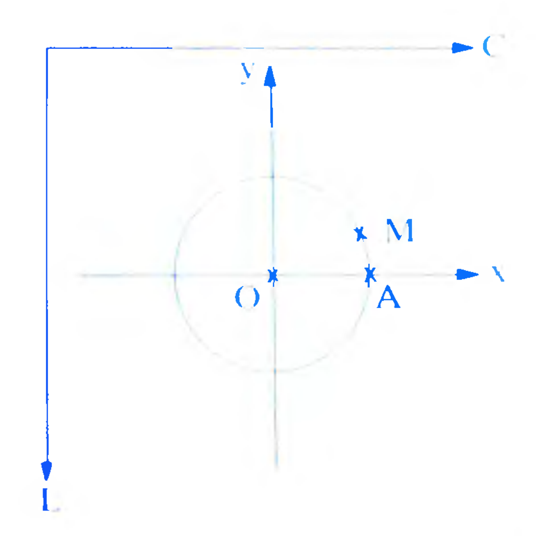
Exemple 5/Cercle

But du programme

Dessiner un cercle du rayon que l'on veut et à l'endroit que l'on veut de l'écran.
Le centre aura pour n° de colonne CØ et pour n° de ligne LØ, le rayon sera R.



Commentaire du programme



Les coordonnées x et y d'un point M du cercle sont :
x = R*cost
y = R*sint
avec t variant de Ø à 2*π.

Comme toujours lorsqu'on trace une courbe, *y* est croissant vers le haut alors que les lignes à l'écran sont croissantes vers le bas. Les n° de colonne et de ligne du point M sont donc :

C = CØ + x   ou   C = CØ + R*cost
L = LØ − y   ou   L = LØ − R*sint

En particulier pour t = Ø, on obtient le point A qui sera le point de départ du dessin :

A { C = CØ + R
    L = LØ

Le programme positionne d'abord le curseur au point A (ligne 6Ø). Puis il calcule la position du point suivant et le relie au point précédent (boucle L.7Ø-11Ø) et ainsi de suite jusqu'à t ≈ 6.3(2*π ≈ 6.28).

Pour voir le rôle du pas de variation de t, essayez les modifications suivantes de la ligne 7Ø :

```
7Ø FOR T = Ø TO 6.3 STEP Ø.5
7Ø FOR T = Ø TO 6.3 STEP 1
7Ø FOR T = Ø TO 6.3 STEP Ø.Ø5
```

Vous pouvez utiliser ce programme pour placer le cercle où vous voulez : il suffit de changer les coordonnées du centre CØ et LØ et vous pouvez modifier son rayon R.

Remarque :
Si vous mesurez le diamètre vertical et le diamètre horizontal du cercle, vous trouverez des longueurs peut-être légèrement différentes. Si vous voulez un "cercle parfait", il faut multiplier le terme R*SIN(T) dans la ligne 9Ø par un coefficient calculé pour égaliser les deux diamètres.

```
10 ' CERCLE
20 CLS
30 COULEUR=3
40 C0=160 : L0=100 : ' CENTRE
50 R=80            : ' RAYON
60 PSET(C0+R,L0),COULEUR
70 FOR T=0 TO 6.3 STEP 0.1
80    C=C0+R*COS(T)
90    L=L0-R*SIN(T)
100   LINE -(C,L),COULEUR
110 NEXT T
```

Exemple 6/Test mémoire visuelle

But du programme

Faire apparaître successivement une vingtaine de mots puis demander d'en citer cinq parmi cette liste.
Les mots sont affichés très rapidement au centre de l'écran les uns à la suite des autres. Quand ils ont tous été vus, il faut en citer cinq. Le programme répond en signalant si le mot cité était dans la liste ou non.
Ce programme est un entraînement stimulant de la perception des mots et de la mémoire immédiate.

Commentaire du programme

Les mots qui vont être affichés sont rangés en DATA à la fin du programme à partir de la ligne 1ØØØ. Il sera ainsi facile de les modifier. Comme l'affichage des mots est très rapide, il importe de ne commencer que lorsque le joueur est prêt et après avoir attendu un temps minimum (lignes 3Ø à 8Ø).

L'affichage proprement dit est constitué d'une boucle (lignes 1ØØ à 18Ø) qui lit le mot, l'écrit au milieu de l'écran, attend un court instant (boucle de temporisation) puis efface et passe au mot suivant.

L'instruction LOCATE en ligne 15Ø est suivie d'un troisième chiffre (le premier indique le n° de colonne, le second le n° de ligne). Quand ce chiffre vaut Ø, il fait disparaître le curseur et quand il vaut 1, le fait réapparaître.
Cette particularité nous est bien utile car le curseur pourrait gêner la lecture des mots. Le curseur réapparaît par la suite (ligne 22Ø).

Le contrôle des mots donnés par le joueur est constitué par une deuxième boucle (lignes 2ØØ à 3ØØ).
Pour chaque mot donné on vérifie qu'il se trouve dans la liste. On indique ensuite le résultat par "juste" ou "faux" et on augmente le score d'une unité dans le cas "juste".
Le score final est indiqué à la fin du texte.

Améliorations

Pour rendre l'exercice plus facile ou plus difficile, il est possible de modifier trois données : la durée de l'affichage pour un mot,

le nombre et la difficultés des mots, le nombre de mots à citer après la lecture.

Ces paramètres ont été mis en variables pour en faciliter la modification :

NMOT	nombre de mots affichés.
T	nombre de boucles de temporisation
NC	nombre de mots demandés en contrôle.

Il faut vérifier toutefois que le nombre de mots affichés NMOT ne dépasse pas le nombre de mots présents en DATA à la fin du programme. Si c'était le cas, il faudrait en rajouter en quantité suffisante.

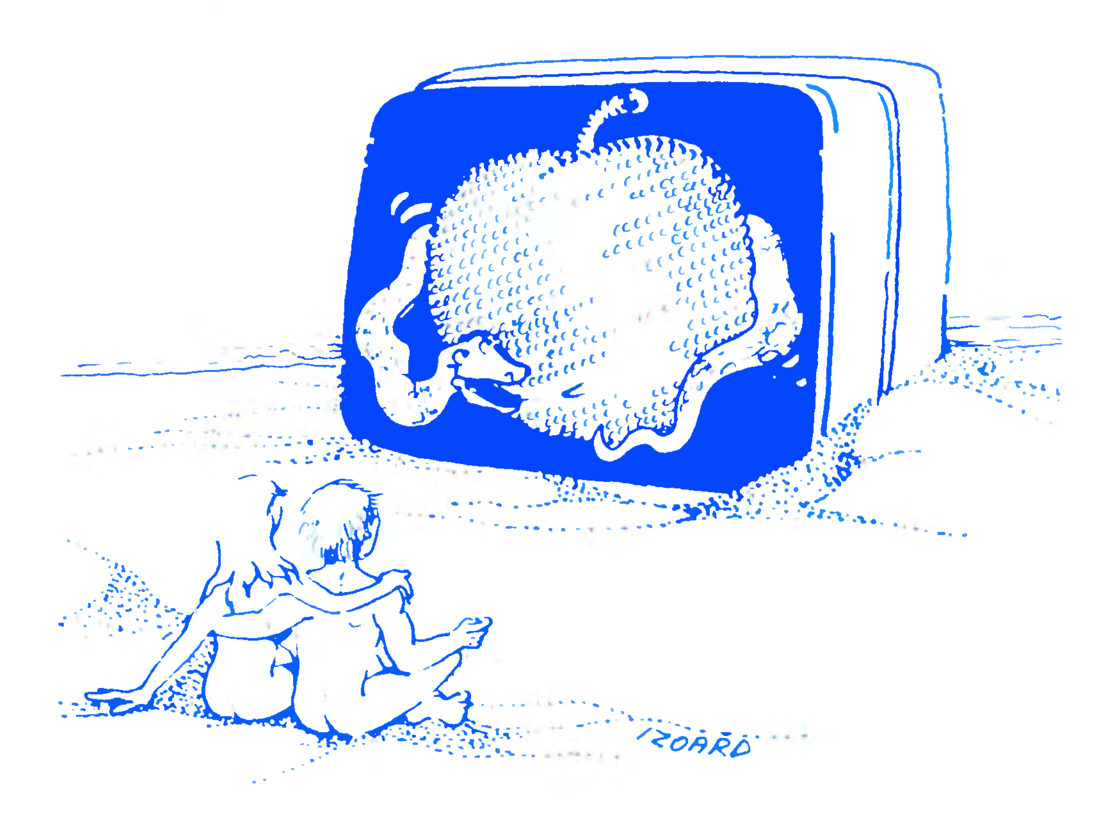
```
10  'TEST MEMOIRE
20  CLS
30  LOCATE 7,10
40  LOCATE 10,7: PRINT "TEST DE MEMOIRE"
50  PRINT "QUAND VOUS SEREZ PRET."
60  PRINT TAB(7) "APPUYEZ SUR ENTREE ";
70  INPUT R$ : CLS
80  FOR W=1 TO 300 : NEXT W
90  '
100 '       AFFICHAGE
110 NMOT=20
120 T=20
130 FOR I=1 TO NMOT
140  READ MOT$
150  LOCATE 16,12,0 :PRINT MOT$
160  FOR W=1 TO T : NEXT W
170  CLS
180 NEXT I
190 '
200 '       CONTROLE
210 NC=5 : SCORE=0
220 LOCATE 5,5,1
230 PRINT "CITEZ";NC;"MOTS DE LA LISTE"
240 FOR I=1 TO NC
250 PRINT I; : INPUT R$
260  RESTORE : N=0
270  IF N<>NMOT THEN 290
280    PRINT "FAUX" : GOTO 330
290  READ MOT$
300  IF R$=MOT$ THEN 320
310  N=N+1 : GOTO 270
320  PRINT "JUSTE" : SCORE=SCORE+1
330 NEXT I
340 PRINT : PRINT "SCORE ";SCORE;"/";NC
350 END
    999 '
1000 DATA "BONJOUR","MAISON","CARTON",
     "ANIMAL","BROUETTE","TELEVISION",
     "PAPIER"
1020 DATA "CRAYON","ORDINATEUR",
     "PISCINE","MONTAGNE","LIGNE",
     "LAMPE","MAGASIN"
1040 DATA "TROTTOIR","PANTALON",JOUET",
     "EPICERIE","GOMME","COULEUR"
```

Exemple 7/Quelques courbes

But du programme

Dessiner des courbes dont on connaît (ou invente) les équations :

x = f(t)
y = g(t)



Cardioïde :

Cette courbe qui tire son nom de sa ressemblance avec un cœur va nous servir ici à dessiner une pomme à l'écran.
Ses équations sont :

x = A(1 – cost)cost
y = B(1 – cost)sint

En changeant A et B vous changerez les proportions verticales et horizontales de la courbe.

Le point de départ de la courbe correspond à t = Ø donc x = Ø et y = Ø. Nous l'avons placé sur la colonne 16Ø et la ligne 1ØØ.

Pour un point quelconque de la courbe, on a donc :
C = 16Ø + X
L = 1ØØ – Y
ou :
C = 16Ø + A(1 – cost)cost
L = 1ØØ – B(1 – cost)sint

```
10  'CARDIOIDE
20  CLS
30  A=40 : B=40
40  PI=3.1416
50  PSET (160,100),2
60  FOR T=0 TO 2*PI STEP 0.1
70     X=A*(1-COS(T))*COS(T)
80     Y=B*(1-COS(T))*SIN(T)
90    C=X+160
100   L=100-Y
110   LINE -(C,L),2
120 NEXT T
```

Pomme

Pour dessiner une pomme à partir de la cardioïde, nous avons d'abord mis la pomme "debout" en inversant les rôles de X et Y :
C = 16Ø + Y
L = 1ØØ – X
Il suffit ensuite de rajouter la queue de la pomme (L.14Ø)

```
10  'POMME
15  SCREEN,0,0
20  CLS
30  A=40 : B=40
40  PI=3.1416
50  PSET (160,100),1
70  FOR T=0 TO 2*PI STEP 0.1
80     X=A*(1-COS(T))*COS(T)
90     Y=B*(1-COS(T))*SIN(T)
100    C=Y+160
110    L=100-X
120    LINE -(C,L),1
130 NEXT T
140 LINE (160,100)-(170,85),2
```

Rosace

Suivant le même principe que pour le cercle ou la cardioïde, vous pourrez tracer un grand nombre de courbes. Par exemple voici une rosace dont l'équation est :

x = K.sin 2t/3 . cost

y = K.sin 2t/3 . sint

Pour t = Ø, on obtient :

x = Ø

y = Ø

Nous avons placé ce point en :

C = 16Ø

L = 1ØØ

```
10 '   ROSACE
20 CLS
30 K=70
40 PI=3.1416
50 PSET (160,100),3
60 FOR T=0 TO 6*PI STEP 0.1
70  X=K*SIN(2*T/3)*COS(T)
80   Y=K*SIN(2*T/3)*SIN(T)
90   C=X+160
100  L=100-Y
110  LINE-(C,L),3
120 NEXT T
```

Lissajous

Autre exemple de courbes : les courbes de Lissajous dont l'équation est :

x = Ksin At

y = Kcos Bt

Le point de départ est :

C = 16Ø

L = 3Ø

Vous pouvez obtenir d'autres courbes en changeant A et B : gardez par exemple A = 2 et essayez B = 2, B = 3, B = 5, B = 7 ...

```
10 'LISSAJOU
20  CLS : K=70
30 A=2 : B=5
40 PI=3.1416
50 PSET (160,30),4
60 FOR T=0 TO 2*PI+0.5 STEP 0.05
70  X=K*SIN(A*T)
80  Y=K*COS(B*T)
90  C=X+160
100 L=100-Y
110 LINE -(C,L),4
120 NEXT T
```

Exemple 8/Horloge

But du programme

Cette horloge affiche au centre de l'écran les lettres H, M, S et juste au-dessous de chaque lettre, les valeurs des heures, minutes et secondes.
Elle se met à l'heure en demandant au début du programme de préciser les valeurs des heures, minutes, secondes.



Commentaire

Le programme utilise trois variables : H pour les heures, M pour les minutes, S pour les secondes.

Une première partie (L.3Ø – 7Ø) demande à l'utilisateur de préciser les valeurs initiales de H, M, S. Il faut entrer le chiffre des secondes S (en appuyant sur ENTREE) lorsque la montre indique le chiffre annoncé (l'horloge démarre immédiatement). L'affichage des lettres H, M, S (L. 9Ø) est valable pour tout le programme.

La troisième partie du programme (L.1ØØ – 19Ø) est une boucle dont la durée totale est de 1 seconde. Au cours de cette boucle :
1 — On affiche les valeurs de H, M, S juste au-dessous des lettres correspondantes grâce à l'instruction :
PRINT USING " # # "
Cette instruction permet d'afficher un nombre avec le chiffre des unités toujours à la place du 2e dièse et celui des dizaines (si le nombre est supérieur à 9) à la place du 1er dièse.
Vous pouvez supprimer temporairement cette instruction pour voir son rôle (en ajoutant une ' après le numéro de ligne).
2 — On attend le temps nécessaire pour que la durée totale de la boucle soit 1 seconde : boucle de temporisation en ligne 13Ø.
3 — On augmente le nombre des secondes d'une unité. S'il atteint 6Ø, on le ramène à Ø et on augmente le nombre des minutes de 1.
Si le nombre des minutes atteint alors à son tour la valeur 6Ø, on le ramène à Ø et on augmente le nombre des heures de 1.
Nous n'avons pas prévu le cas où l'on utilise l'horloge autour de minuit !
Si c'est votre cas, rajoutez dans le programme les 2 lignes qui ramènent H à Ø lorsque H atteint 24.

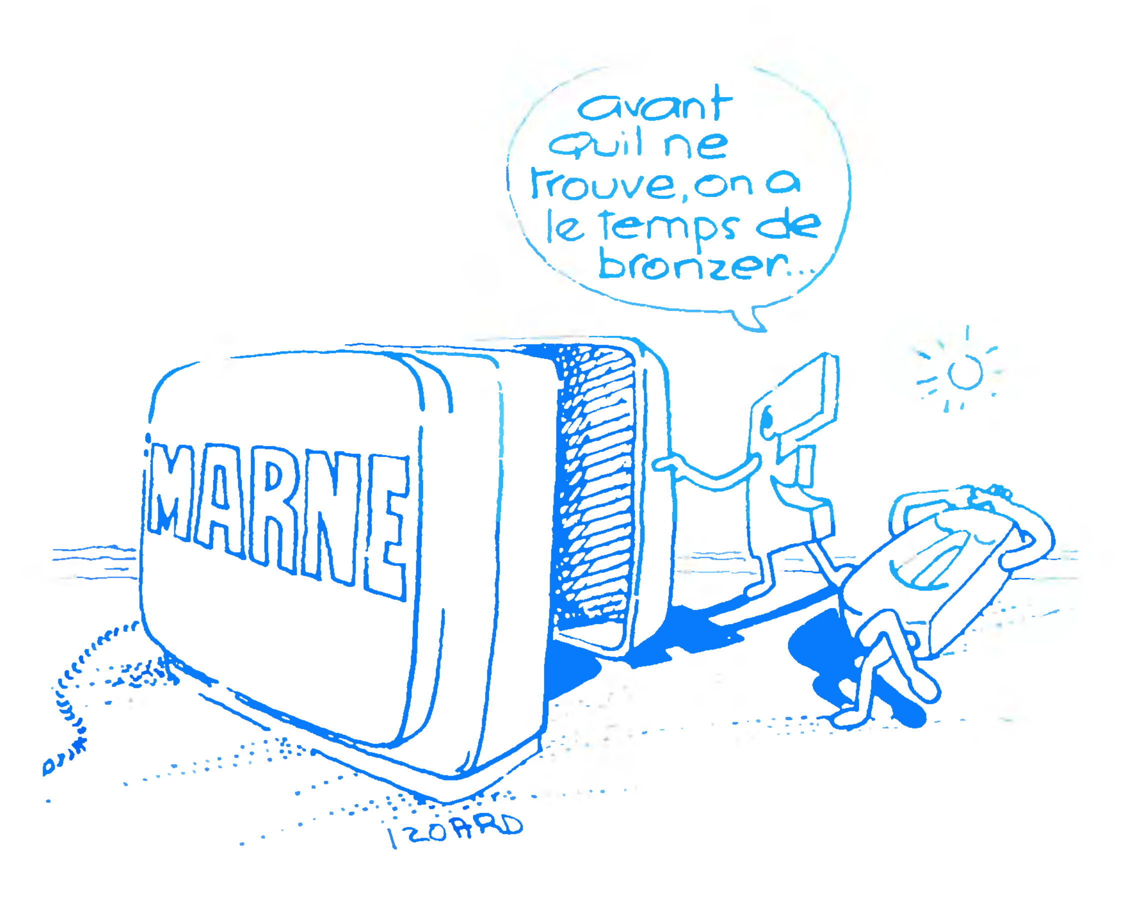
```
10 ' HORLOGE
20 CLS
30 LOCATE 7,4 : PRINT "MISE A L'HEURE"
40 PRINT
50 INPUT "-HEURE";H
60 INPUT "-MINUTES";M
70 INPUT "-SECONDES";S
80 CLS : ATTRB 1,1
90 LOCATE 14,10,0 : PRINT"H  M  S"
100 ' AFFICHAGE DE L'HEURE
110 LOCATE 10,14,0
120 PRINT USING " ##";H;M;S
130 FOR W=1 TO 425 : NEXT W
140 S=S+1
150 IF S<>60 THEN 100
160 S=0 : M=M+1
170 IF M<>60 THEN 100
180 M=0 : H=H+1
190 GOTO 100
200 END
```

Exemple 9/Jeu du mot de 5 lettres

But du programme

Dans ce jeu qui se joue à 2, chaque joueur choisit un mot français de 5 lettres que son adversaire va chercher à deviner.
Pour cela chacun propose à tour de rôle un mot de 5 lettres et l'autre lui répond le nombre de lettres qui sont "bien placées" par rapport au mot cherché.
Le premier qui a trouvé le mot de son adversaire et qui a donc 5 lettres "bien placées" a gagné.

Ici vous allez jouer contre l'ordinateur, c'est-à-dire que vous allez chercher à deviner un mot que l'ordinateur a choisi (au hasard dans une liste de mots donnés).
Le programme doit donc comparer chaque mot proposé par le joueur au mot choisi et répondre au joueur le nombre de lettres bien placées.
Lorsque les 5 lettres sont bien placées, le programme donne un message qui annonce au joueur qu'il a gagné.



Commentaire

— Rappel de la règle du jeu
Le programme que nous vous proposons rappelle d'abord la règle du jeu (L.3Ø – 8Ø).
L'instruction
PRINT TAB(9) "JEU DU MOT DE 5 LETTRES"
permet d'afficher le titre du jeu environ au milieu de l'écran : la 1re lettre "J" est positionnée sur le 9e caractère de la ligne.

— Stockage des mots à deviner
La liste des mots mis en réserve par le programme est stockée dans l'instruction DATA (L.5ØØ). Vous pourrez évidemment supprimer certains mots ou en ajouter de nouveaux.

— Tirage au sort du mot à deviner (L.8Ø-15Ø)
Tant que le joueur n'appuie pas sur la touche [ENTREE], la variable Z reçoit à chaque tour un nouveau nombre dans la liste des nombres aléatoires (L.8Ø-11Ø).
L'appui sur [ENTREE] au bout d'un temps quelconque arrête la boucle à un nombre différent et X est bien un nombre au hazard entre 1 et 2Ø.

— Choix du mot à deviner
Le mot à deviner est représenté par 5 tirets sur la 1re ligne de l'écran (L.16Ø)
Cette 1re ligne est réservée pour toute la suite du jeu par l'instruction
CONSOLE 1
Même si le joueur propose 5Ø mots, cette ligne ne s'effacera pas : le déroulement des lignes à l'écran commencera seulement à la 2e ligne (de même CONSOLE 2 réserve les deux premières lignes).
En fin de programme (L.32Ø), l'instruction CONSOLE Ø permet de revenir à un déroulement normal à l'écran où aucune ligne n'est réservée.
A tout instant pendant le jeu, le joueur peut remonter le curseur sur l'un des tirets et le remplacer par la lettre qu'il pense avoir devinée, puis redescendre le curseur à sa place initiale. Ceci, simplement utile pour "visualiser" ses hypothèses, n'est pas indispensable et n'intervient pas dans le programme.

— Comparaison entre le mot proposé et le mot à deviner (L.18Ø – 27Ø)

Après avoir affiché les 5 tirets, le programme affiche sur la ligne suivante un point d'interrogation qui invite le joueur à proposer un mot : R$.

Le nombre de coups, c'est-à-dire de mots proposés est comptabilisé dans la variable L.
Le nombre de lettres bien placées est T.
Les lignes 2ØØ – 22Ø préviennent le joueur qu'il a proposé un mot de moins de 5 lettres (ou plus) et lui demandent de recommencer.
La boucle 25Ø – 27Ø compare chaque lettre du mot proposé R$ avec la lettre correspondante du mot à chercher X$ et totalise le nombre T de lettres bien placées.

— Affichage du nombre T
Pour afficher T à la suite du mot R$, sur la même ligne, on remonte le curseur d'une ligne avec l'instruction (L.2ØØ) :
PRINT CHR$(11);
Cette instruction équivaut à appuyer en mode direct sur la touche [↑].

— Prévenir le joueur lorsqu'il a gagné
En fait dès que T = 5, le joueur sait qu'il a gagné. Mais on peut le gratifier un peu avec :
• quelques notes de musique
• un affichage en gros caractères rouges de "GAGNE EN I COUPS".
L'instruction
ATTRB 1, 1 double la hauteur et la largeur des caractères
L'instruction
COLOR 1 donne la couleur 1 (rouge) aux caractères.

Remarques

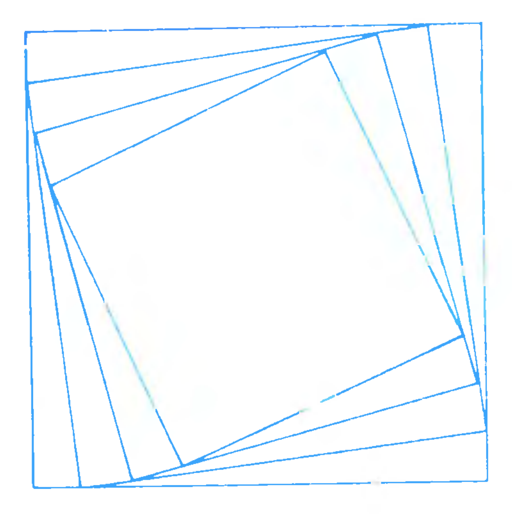
— Si le joueur abandonne en cours de partie avec [CNT] C, donc sans avoir trouvé le mot à deviner, il faut faire ensuite en mode direct :
CONSOLE Ø
pour que le déroulement se fasse normalement à l'écran.
— Le jeu est prévu pour fonctionner en mode majuscule : si vous donnez la bonne réponse en minuscule, l'ordinateur vous répondra Ø !

```
10 'MOT
20 CLS
30 PRINT TAB(8)
   "- JEU DU MOT DE 5 LETTRES -"
40 PRINT : PRINT
50 PRINT "VOUS ALLEZ DEVINER UN MOT
   DE CINQ LETTRES."
60 PRINT " PROPOSEZ DES MOTS ET JE VOUS
DIRAI"
70 PRINT : PRINT " LE NOMBRE DE LETTRES
BIEN PLACEES."
80 PRINT : PRINT"APPUYEZ SUR ENTREE"
90 A$=INKEY$
100 Z=RND
110 IFA$="" THEN 90
120 X=INT(20*RND)+1
130 FOR I=1 TO X
140  READ X$
150 NEXT I
160 CLS
170
180 CONSOLE 1
190 PRINT " -----"
200 I=0 : ' NOMBRE DE COUPS
210  INPUT R$
220  T=0 : ' NOMBRE DE LETTRES BIEN
             PLACEES
230  PRINT TAB(8) CHR$(11);
240  IF LEN(R$)=5 THEN 270
250  PRINT" DONNEZ UN MOT DE 5 LETTRES"
260  GOTO 210
270  I=I+1
280  FOR K=1 TO 5
290   IF MID$(R$,K,1)=MID$(X$,K,1)
      THEN T=T+1
300  NEXT K
310  PRINT T
320  IF T<>5 THEN 210
330 PLAY "SIMISO" : ATTRB 1,1 : COLOR 1
340 PRINT : PRINT "GAGNE EN";I;"COUPS
350 CONSOLE 0 : COLOR 7
360 END
500 DATA  NICHE,SAPIN,TEMPS,LIVRE,FUMEE,
          ECOLE,CATCH,FEMME,MARIN,STYLO,
          NUAGE,YATCH,FORET,OLIVE,CHIEN,
          PENDU,LACET,PORTE,OCEAN,ANGLE
```

Exemple 10/Carrés imbriqués

But du programme

Dessiner une succession de carrés imbriqués de la manière suivante :

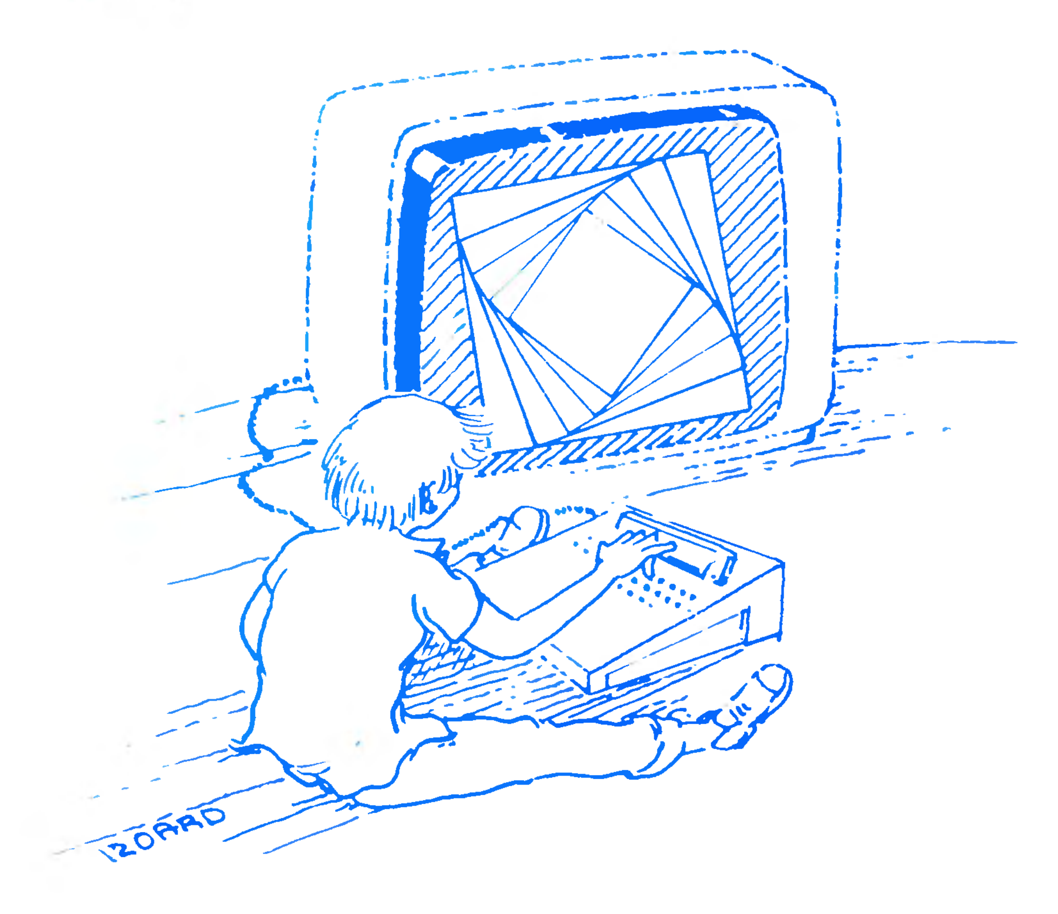


Commentaire du programme

En numérotant les sommets d'un carré 1, 2, 3, et 4, les coordonnées (colonne et ligne) seront notées :
C(1), L(1)
C(2), L(2)
C(3), L(3)
C(4), L(4)
Pour passer d'un carré à un autre, il faut déplacer chaque sommet d'une certaine quantité sur le côté qui le lie au sommet suivant.
Le sommet 1 glisse légèrement vers le sommet 2, le 2 vers le 3, ...
Nous avons choisi de les déplacer du dixième du côté (K = .1).
Ce changement est fait par une boucle (L.13Ø – 16Ø).
Pour effectuer le dessin, il est utile d'introduire un sommet numéro 5, situé au même endroit que le numéro 1. Une boucle trace 5 segments successifs : le premier permet de se positionner au nouvel emplacement du sommet 1, les quatres autres pour tracer le carré proprement dit (L.1ØØ – 12Ø).
Il suffit de répéter l'exercice précédent autant de fois qu'on le désire (ici NB = 3Ø).

C'est l'objet de la boucle principale (L.9Ø – 18Ø).
Attention, le dessin n'utilise que LINE – (,). Il ne faut pas oublier de fixer le point de départ (L. 8Ø).

```
10 ' CARRES
20 CLS : K=.1
30 C(1)=80 : L(1)=20
40  C(2)=80 : L(2)=180
50  C(3)=240 : L(3)=180
60  C(4)=240 : L(4)=20
70  C(5)=C(1) : L(5)=L(1)
80  PSET(C(1),L(1))
90  FOR NB=1 TO 30
100   FOR I=1 TO 5
110     LINE -(C(I),L(I))
120   NEXT I
130   FOR I=1 TO 4
140     C(I)=C(I)+(C(I+1)-C(I))*K
150     L(I)=L(I)+(L(I+1)-L(I))*K
160   NEXT I
170   C(5)=C(1) : L(5)=L(1)
180 NEXT NB
190 END
```



Exemple 11/Télécran

But du programme

Dessiner à l'écran avec les 4 touches : [↑] [↓] [→] [←]
L'appui sur une de ces flèches trace un petit trait à l'écran dans le sens de la flèche (de longueur 1Ø par exemple).

Fonction nouvelle à utiliser : INPUT$( )

Cette fonction permet d'entrer en mémoire un ou plusieurs caractères sans terminer par [ENTREE] .
Le petit programme suivant vous montre comment on peut l'utiliser :

```
NEW
10 A$=INPUT$(1)
20 PRINT "TOUCHE APPUYEE : ";A$
RUN
```

La ligne 1Ø indique que l'ordinateur va attendre qu'une touche soit appuyée au clavier. Le caractère correspondant à la touche se trouve alors dans A$. A la différence de INKEY$ qui n'attend pas, le programme ne continue que lorsque le nombre de caractères attendus (ici 1) a été entré au clavier.
Si vous appuyez sur une touche [←] ou [INS] , la ligne 2Ø est exécutée mais A$ n'est pas vide ; ajoutez la ligne suivante :

```
30 PRINT"CODE ASCII DE CETTE TOUCHE : "
   ;ASC(A$)
```

Vous verrez ainsi les codes ASCII des touches du clavier.

Commentaire du programme

On demande d'abord au joueur en quelle position de l'écran il désire placer le spot au départ.
Cette ligne 3Ø peut être modifiée pour fixer le point de départ par exemple au milieu de l'écran :

```
3Ø X = 16Ø : Y = 1ØØ
```

Pour éliminer le curseur de l'écran, on l'a positionné en un point

quelconque (1, 1) et, ce qui est important, on l'a rendu invisible en ajoutant un troisième chiffre Ø dans l'instruction LOCATE :

LOCATE 1, 1, Ø

Le cadre de l'écran dans lequel va s'inscrire le dessin est dessiné en ligne 6Ø.

A chaque fois que le joueur appuie sur une touche le programme décrit la boucle 8Ø – 16Ø :
— la ligne 8Ø affecte le caractère appuyé à la variable A$
— les lignes 9Ø à 12Ø testent laquelle des quatres flèches a été appuyée par le joueur (le code ASCII de ces touches est en effet 8, 9, 1Ø, 11) et modifie en conséquence la coordonnée C ou L pour placer le point suivant.
— les lignes 13Ø et 14Ø ramènent les variables C et L à Ø si le joueur "fait sortir" le spot du cadre (vers le haut ou vers la gauche). Il est inutile de faire de même si le joueur "fait sortir" le sport vers la droite ou vers le bas puisque l'ordinateur accepte des n° de ligne ou de colonne supérieurs aux valeurs limites (319 pour les colonnes et 199 pour les lignes).

```
10 ' TELECRAN
20 CLS
30 INPUT "POINT DE DEPART (C,L)";C,L
40 INPUT "COULEUR ";COULEUR
50 CLS : LOCATE 1,1,0
60 BOX (0,0)-(320,200),COULEUR
70 PSET (C,L),COULEUR
80      A$=INKEY$
90      IF A$=CHR$(8)    THEN C=C-10
100     IF A$=CHR$(9)    THEN C=C+10
110     IF A$=CHR$(10)   THEN L=L+10
120     IF A$=CHR$(11)   THEN L=L-10
130     IF C<0 THEN C=0
140     IF L<0 THEN L=0
150     LINE -(C,L),COULEUR
160     GOTO 80
```

Exemple 12/Caractère personnalisé

But du programme

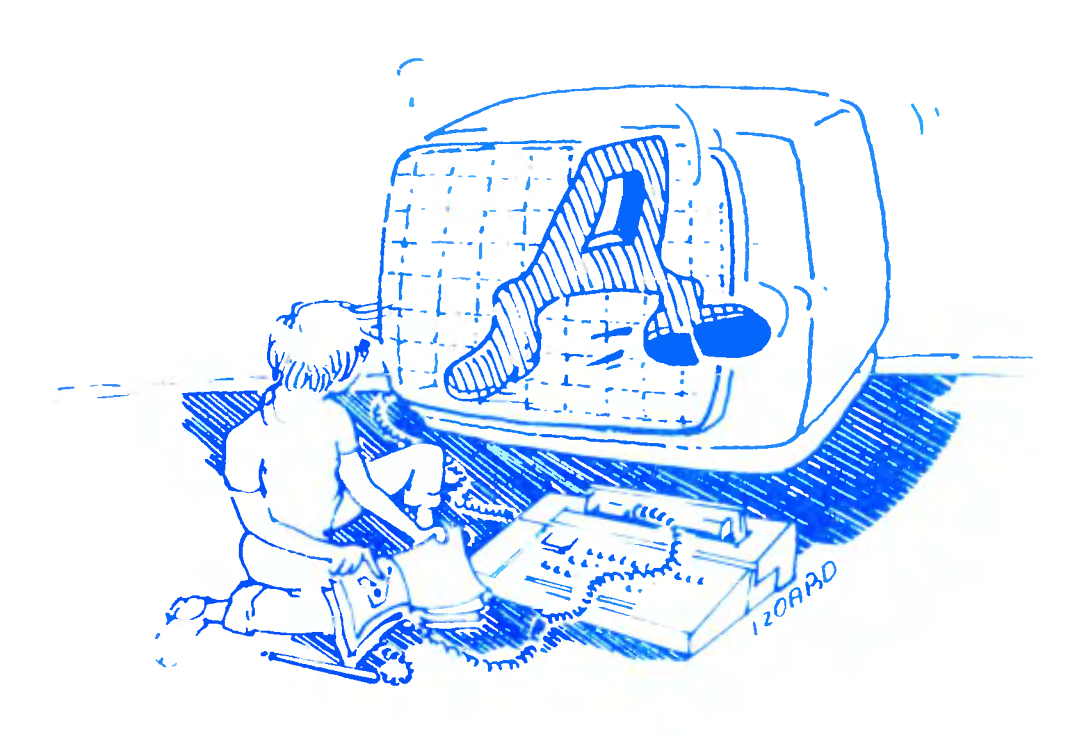
Ce programme ne doit pas être considéré comme un exemple de programmation mais plutôt comme un programme que vous pourrez utiliser tel quel. Il est en effet un peu plus compliqué que les précédents et il se justifie surtout par l'usage que vous pourrez en faire par la suite.

Son but est de vous donner la possibilité de définir un caractère (8 × 8) à votre gré.

Comme il est indiqué dans l'Annexe 2, il est possible de définir point par point un certain nombre de caractères. Ce programme vous permettra de dessiner directement à l'écran sur une grille représentant le caractère agrandi, le caractère que vous voulez définir.

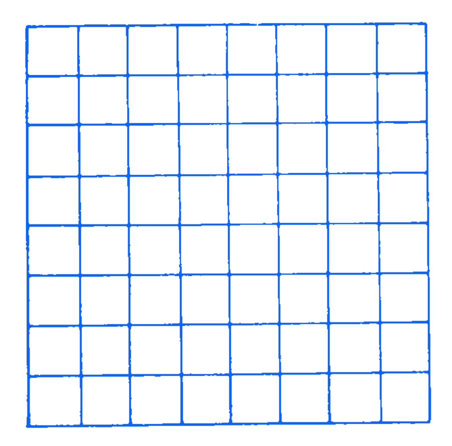
A côté de la grille s'affiche en vraie grandeur (c'est-à-dire 8 × 8) le caractère au fur et à mesure de sa composition.

Quand celle-ci est terminée, le programme affiche l'instruction que vous devrez placer dans un programme pour dessiner ce caractère.



Commentaire du programme

Le programme ne permet de définir qu'un seul caractère à la fois.
Le nombre de caractères définis par l'utilisateur (ici 1) doit être indiqué par une instruction CLEAR (ligne 2Ø). Précisons au passage la syntaxe de cette instruction :
CLEAR n1, n2, n3
n1 fixe l'emplacement réservé aux chaînes de caractères
n2 fixe l'emplacement réservé aux sous-programmes assembleur (à voir plus tard)
n3 est le nombre de caractères définis par l'utilisateur
Ici nous fixons seulement le nombre de caractères n3 :
CLEAR ,,1
Le caractère agrandi est représenté au moyen d'une grille de 8 sur 8, soit au total 64 carrés élémentaires correspondant chacun à un point.



On noircit avec le crayon optique le carré correspondant à un point "allumé" (on considère que le dessin se fait en noir sur fond clair).
Chaque point "allumé" doit être mémorisé. La grille complète est donc mémorisée dans un tableau à deux dimensions CAR (8, 8).
Chaque élément de ce tableau vaut Ø s'il est éteint et 1 s'il est allumé.

Le dessin de cette grille est fait dans un sous-programme en ligne 8ØØ.

Notez que le fond de couleur jaune est marqué avec un numéro de couleur négatif (ligne 81Ø). En effet, une couleur qui est

utilisée en fond (parce qu'on écrira ou dessinera par dessus) est notée avec un nombre négatif.

Pour ne pas avoir à répéter plusieurs fois les mêmes nombres constants, et pour pouvoir les changer au besoin, les coordonnées des deux points opposés de la grille ont été mis dans des variables : 1er point (MI, MI), 2e point (MA, MA).

La définition du caractère va se faire uniquement avec le crayon optique. En appuyant sur l'un des carrés élémentaires, on le noircit. Quand c'est terminé, il suffit d'appuyer à droite sur une case marquée "FIN".

Le programme comprend deux parties principales :
— composition du caractège (lignes 200 – 290)
— fin de la composition (lignes 300 – 360)

Composition du caractère

A partir de la colonne X et de la ligne Y pointées par le crayon optique, on calcule la position C, L du carré qui a été visé (L. 210 – 220), sachant que chaque case a un côté de 16.
Si ce carré a déjà été marqué, il n'y a rien de plus à faire (L.230). Sinon, il faut noter qu'il a été visé et noircir la boîte correspondante (L. 240 – 250).

Il faut ensuite redéfinir le caractère et l'afficher en grandeur normale à côté de la grille. La définition se fait après avoir recalculé le nombre correspondant à la ligne qui a été touchée : NL(L) (ligne 260).

Fin de la composition

L'instruction de définition du caractère est affichée en haut de l'écran. Il ne vous restera plus qu'à la recopier dans le ou les programmes où ce caractère doit intervenir.
Pour vérifier, en dessinant un caractère en forme de carreau on obtient :



DEFGR$(x) = 16, 56, 124, 254, 124, 56, 16, 0

```
10  ' COMPOSITION D'UN CARACTERE
20  DIM CAR(8,8) : CLEAR,,1
```

```
30  SCREEN 0,7,0 : CLS
40  MI=48 : MA=175
50  PRINT TAB(5) "DESSIN D'UN CARACTERE"
60  GOSUB 800 :' GRILLE
100 INPUTPEN X,Y
110 IF X=-1 THEN 100
120 IF X>MI AND X<MA AND Y>MI AND Y<MA
    THEN 200
130 IF X>280 AND Y>160 THEN 300
140 GOTO 100
200 ' POINT VISE
210 C=INT((X-MI)/16)+1
220 L=INT((Y-MI)/16)+1
230 IF CAR(C,L)=1 THEN 100
240 CAR(C,L)=1
250 BOXF (MI+16*(C-1),MI+16*(L-1))-
         (MI+16*C,MI+16*L)
260 NL(L)=NL(L)+2^(8-C)
270 DEFGR$(0)=NL(1),NL(2),NL(3),NL(4),
              NL(5),NL(6),NL(7),NL(8)
280 LOCATE 32,10 : PRINT GR$(0)
290 GOTO 100
300 ' FIN DU DESSIN
310 LOCATE 1,1
330 FOR I=1 TO 8
340   PRINT NL(I);
350 NEXT I
360 END
800 ' TRACE DE LA GRILLE
810 BOXF(MI,MI)-(MA,MA),-4
820 FOR I=0 TO 8
830   LINE (MI,MI+16*I)-(MA,MI+16*I)
840   LINE (MI+16*I,MI)-(MI+16*I,MA)
850 NEXT I
860 BOXF (280,160)-(320,200),-4
870 BOX  (280,160)-(320,200)
880 LOCATE 36,18 : PRINT "FIN"
890 LOCATE 28,8  : PRINT "CARACTERE"
900 RETURN
```

ANNEXES

1 — Comment enregistrer et relire vos programmes sur cassettes ?

2 — Le jeu de caractères

3 — Un petit coup d'œil sur la mémoire

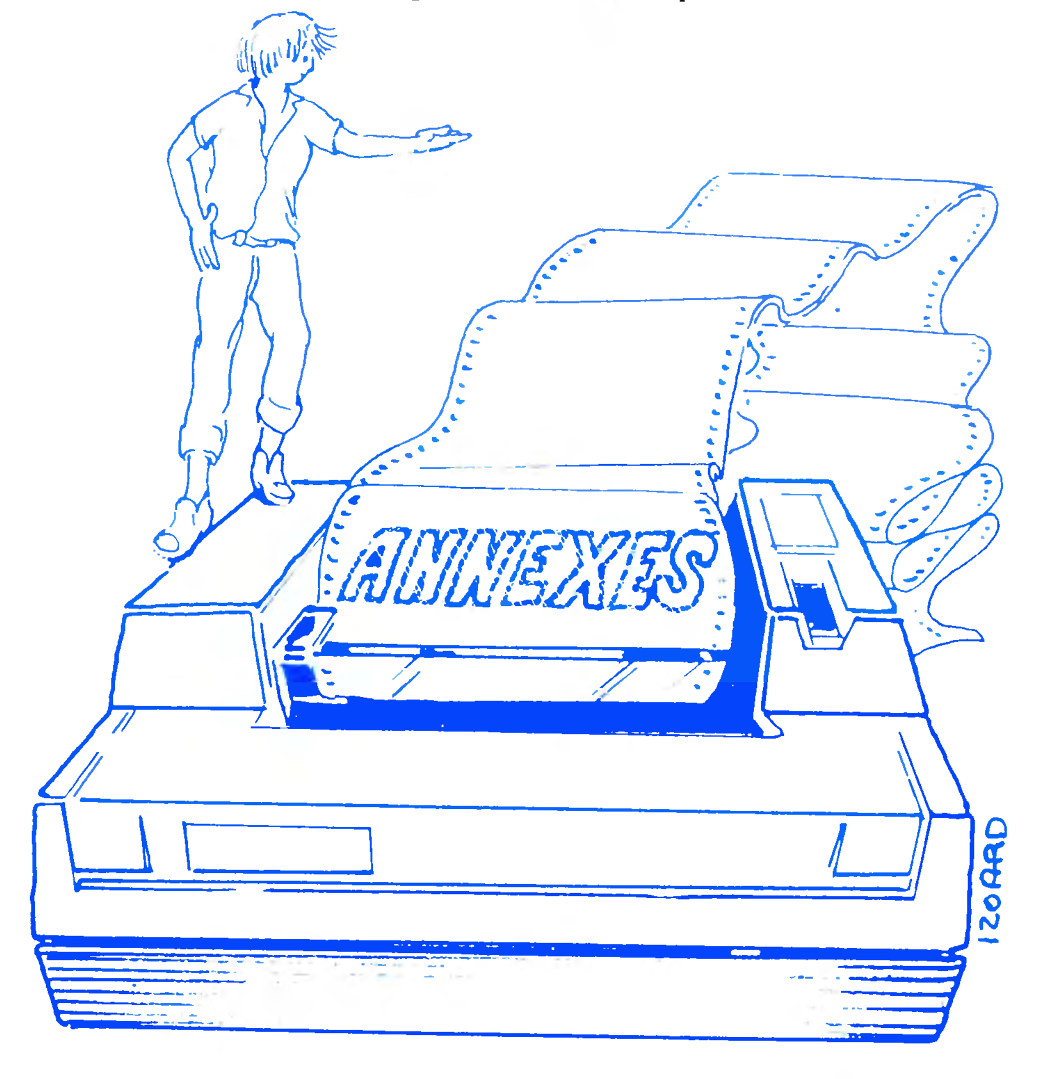
4 — Fonctions mathématiques

5 — Liste des mots réservés

6 — Liste des principaux codes d'erreur

7 — Résumé des instructions

8 — Utilisation de l'imprimante et manipulation des fichiers



Annexe I/Comment enregistrer et relire vos programmes sur cassettes ?

L'ordinateur a une bonne mémoire, mais celle-ci a un gros défaut, elle s'efface lorsqu'on l'éteint. Au début, cela n'est pas très gênant, mais dès que vous avez fait un petit programme amusant, vous avez envie de le conserver.
Il faut alors le recopier sur une feuille à la main, ou l'imprimer sur l'imprimante (si vous en avez une) et le retaper à nouveau au clavier pour le réutiliser.

Une bonne solution pour éviter cela : le magnétophone à cassettes. C'est une mémoire plus vaste et plus stable qui, si l'on en prend soin, peut se conserver très longtemps.



L'enregistrement

Le principe de l'enregistrement est simple : il consiste à écrire sur la cassette le texte du programme non pas parlé (nous n'y sommes pas encore), mais codé par des fréquences audibles. Nous allons donc recopier le programme qui est actuellement en mémoire sur cassette, comme on recopie une chanson.
Pour le faire, il faut d'abord brancher la prise du magnétophone

et le connecter au microordinateur grâce au cordon qui porte une prise DIN.

Introduisez une cassette vierge. Vous constatez que les touches de déroulement rapide avant [▶▶] et arrière [◀◀] fonctionnent mais que la touche [▶] est sans action. En effet, elle est sous le contôle de l'ordinateur.

Rembobinez la cassette jusqu'au début (pas trop tout de même, il est inutile d'essayer d'écrire sur la bande amorce).

Puis enfoncez simultanément les touches [●] et [▶] comme pour enregistrer de la musique.

La cassette est bloquée : l'enregistrement ne commencera que lorsque l'ordinateur en donnera l'ordre.

Votre programme est dans l'ordinateur, sinon achevez de le taper.

La commande de BASIC qui va permettre d'écrire sur la cassette est SAVE ("sauve" ou "sauvegarde"). Mais pour s'y reconnaître parmi les divers programme que vous enregistrerez, il va falloir donner un nom au programme à enregistrer.

Supposons que vous vouliez conserver le dernier programme du chapitre VII, celui qui dessine des drapeaux. Vous lui donnerez le nom "DRAPEAU" et la commande d'enregistrement s'écrit :

SAVE "DRAPEAU"

Le nom du programme s'écrit entre guillemets, comme dans l'instruction PRINT .

Immédiatement après que vous ayez tapé cette ligne, le curseur se fige, immobile, et le moteur du magnétophone se met en marche. Quelques secondes plus tard, le moteur s'arrête et en même temps l'ordinateur répond "OK".
Ça y est, le programme est enregistré.
Et bien maintenant, vérifions.

La lecture

Tapez :

NEW

La mémoire est nettoyée. Vérifiez-le en tapant :

LIST

Il n'y a plus de programme en mémoire. Nous allons charger celui qui vient d'être enregistré.

Rembobinez la cassette jusqu'au début. Enfoncez la touche [▶].
La commande BASIC qui permet de lire un programme enregistré est LOAD (en anglais : "charger") suivi du nom du programme cherché.
Ici vous devez donc taper :
LOAD "DRAPEAU"
Le moteur se met en marche et à l'écran s'affiche :
Searching (je cherche),
puis lorsque le début du programme est trouvé :
Found : DRAPEAU.BAS (trouvé : DRAPEAU.BAS)
Attendez encore quelques instants :
OK
Il a fini de lire le programme cherché.

Tapez :
LIST
C'est bien cela : le programme a été rentré en machine à partir de la cassette.

Nous venons de décrire un bon fonctionnement de l'ensemble. Vous avez peut-être remarqué un grésillement dans le haut-parleur au moment de la lecture du programme : il correspond au codage audible du texte des instructions. C'est bon signe.

Mais si vous n'avez rien entendu et que, au bout d'un certain temps, l'ordinateur n'a toujours pas affiché :
Found : DRAPEAU.BAS
ne le laissez pas chercher plus longtemps.
Arrêtez la lecture de la bande avec le bouton de réinitialisation : en effet les touches du clavier sont neutralisées.

Le mieux est de reprendre le tout au début, y compris l'enregistrement.
Vérifiez que le connecteur DIN est bien enfoncé.
Et quand vous rembobinez la bande jusqu'au début, faites la dérouler un peu afin d'éviter d'enregistrer sur la bande amorce.
Cela doit marcher.

Comment gérer vos cassettes

Vous avez constaté que l'enregistrement d'un programme sur cassette est très rapide et qu'il n'a pas occupé une grande place sur la bande.



Avec une cassette de 6Ø minutes, il sera donc possible d'enregistrer beaucoup de programmes.

Mais attention, il faudra s'y retrouver.
Repérez donc la position de chaque programme à l'aide du numéro donné par le compteur et n'hésitez pas à les espacer les uns des autres, de 1Ø en 1Ø par exemple (ou de 2Ø en 2Ø s'ils sont plus gros).
Tenez à jour sur une feuille la liste des programmes enregistrés avec leurs positions.

Pour les relire rapidement, il vous suffira alors de positionner la cassette un peu avant le début du programme, l'ordinateur le trouvera tout de suite.

Donnez des noms significatifs à vos programmes. Vous les reconnaîtrez plus facilement et bien sûr, évitez de donner le même nom à deux programmes différents.
Si vous faites plusieurs versions d'un même programme donnez leur des numéros : DRAPEAU1, DRAPEAU2, DRAPEAU3 ...
(le nom d'un programme ne peut dépasser 8 caractères).
Si vous avez perdu le répertoire de votre cassette, voici un moyen de retrouver ce qu'il y a dessus :
cherchez à lire un programme qui n'existe pas en commençant par le début de la cassette.
Par exemple :
LOAD "X"

L'ordinateur va afficher :
Searching
et à chaque programme rencontré, par exemple DRAPEAU, qui n'est pas le programme cherché :
Skip : DRAPEAU.BAS
Vous obtiendrez ainsi tous les noms des programmes contenus dans la cassette. A la fin arrêtez la lecture avec la touche de réinitialisation.

Pour effacer la cassette

Il n'est pas recommandé d'enregistrer un programme sur une portion de bande déjà enregistrée : ceci est une différence avec l'enregistrement de musique.
Il faut au préalable effacer la cassette sur une partie débordant largement la portion où l'on désire enregistrer. Le plus simple est de l'effacer complètement.

Pour effacer la cassette, rembobinez-la jusqu'au début, débranchez le cordon qui relie le magnétophone à l'ordinateur et appuyez sur les touches [●] et [▶] pour enregistrer "à vide". Lorsque toute la cassette a ainsi été effacée, vous pouvez la réutiliser pour enregistrer de nouveaux programmes.

Si vous tenez vraiment à écrire au milieu d'une cassette partiellement enregistrée, par exemple à la place d'un autre programme, voici ce que vous pouvez faire :
— déroulez la cassette jusqu'à l'emplacement désiré et mieux, un peu avant.
— enfoncez les touches d'enregistrement [●] et [▶] sur le magnétophone.
— tapez au clavier :
MOTORON
le magnétophone se met en marche et enregistre à vide.
— tapez la commande :
SAVE "TOTO"
(si vous voulez que votre programme s'appelle TOTO)
— c'est tout.

Cette solution aura l'avantage de "nettoyer" un peu la cassette avant l'enregistrement du programme proprement dit.

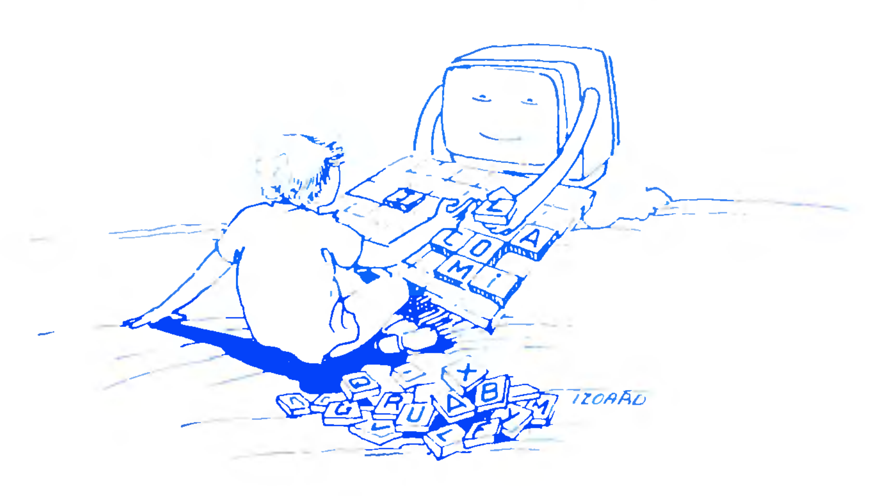
Annexe 2/Le jeu de caractères

Le code ASCII

Les caractères entrés au clavier ou affichés à l'écran (signes, lettres, chiffres) sont présents dans la mémoire sous forme codée. Comme la mémoire est organisée en octets (ensembles de huit bits consécutifs), chaque caractère est rangé dans un octet qui a une certaine valeur.

Un mot de 1Ø lettres par exemple ORDINATEUR est ainsi conservé en mémoire dans 1Ø octets consécutifs.

Parmi les nombreuses possibilités de faire correspondre un nombre à chaque caractère, un code standard a été retenu sur la majeure partie des ordinateurs, il s'agit du code ASCII (American Standard Code for Information Interchange). C'est celui qui est utilisé par le TO7.
Il associe à chaque caractère un nombre compris entre Ø et 127.



Voyons un moyen pratique de retrouver ce code. Il existe tout simplement une fonction BASIC qui donne le code ASCII d'un caractère : cette fonction s'appelle ASC.

```
?ASC("A")
65
```

Le numéro de code de la lettre A est donc 65.

```
?ASC("B")
66
```

Le numéro de code de la lettre B est donc 66.

```
?ASC("2")
5Ø
```

Le numéro de code du chiffre 2 est donc 5Ø.
?ASC("?")
63
Le numéro de code du point d'interrogation est donc 63.
Même le "blanc" a un numéro de code :
?ASC(" ")
32

Vous trouverez plus loin la liste complète des codes mais si vous voulez tester n'importe quel caractère, voici un petit programme qu vous en donnera immédiatement le code :

```
10 PRINT "TAPEZ LE CARACTERE :"
20 A$=INKEY$
30 IF A$="" THEN 20
40 PRINT ASC(A$)
```

A l'inverse, il est tout aussi possible d'obtenir un caractère dont on fournit le code ; c'est le rôle de la fonction CHR$:
?CHR$(65)
A
?CHR$(66)
B
?CHR$(32)
(vous ne voyez rien car l'ordinateur a imprimé un blanc.)
?CHR$(63)
?
Cette fonction permet d'obtenir des "effets" qui ne sont pas à proprement parler des caractères :
?CHR$(7)
l'effet produit est le "BEEP" sonore
?CHR$(8)
le curseur se déplace vers la gauche (c'est donc la flèche [←])
?CHR$(9)
" " la droite (" [→])
?CHR$(1Ø)
" " le bas (" [↓])
?CHR$(11)
" " le haut (" [↑])
?CHR$(3Ø)
" " le coin en haut à gauche de l'écran ([⊐])

Cette fonction a été utilisée dans l'exemple n° 11.

Code ASCII

code	caractère
2	touche STOP
8	flèche ←
9	flèche →
1Ø	flèche ↓
11	flèche ↑
12	touche RAZ
13	touche ENTREE
22	touche ACC
28	touche INS
29	touche EFF
3Ø	touche ⊐
32	barre espacement
33	!
34	’’
35	#
36	$
37	%
38	&
39	’
4Ø	(
41	)
42	*
43	+
44	,
45	–
46	.
47	/
48	Ø
49	1
5Ø	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	:
59	;
6Ø	<
61	=
62	>
63	?
64	a
65	A
66	B
67	C
68	D
69	E
7Ø	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
8Ø	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y
9Ø	Z
91	[
92	\
93	]
94	^
95	_
96	—
97	a

98	b	113	q
99	c	114	r
1ØØ	d	115	s
1Ø1	e	116	t
1Ø2	f	117	u
1Ø3	g	118	v
1Ø4	h	119	w
1Ø5	i	12Ø	x
1Ø6	j	121	y
1Ø7	k	122	z
1Ø8	l	123	{
1Ø9	m	124	|
11Ø	n	125	}
111	o	126	—
112	p	127	■

Les accents et la cédille



Le code ASCII contient donc toutes les lettres, majuscules et minuscules, les chiffres, les signes d'opération et de relation, la ponctuation, et les signes spéciaux comme $, %, ...
Toutefois on n'y trouve pas les minuscules accentuées ou le "ç".

En effet ces caractères sont représentés dans l'ordinateur par plusieurs octets.
Pour chaque minuscule accentuée, il faut trois octets contenant successivement :

1 — le code "minuscule accentuée" : 22

2 — le code de l'accent : 65 pour l'accent grave

65	pour l'accent	grave
66		aigu
67		circonflexe
72	le tréma	
75	la cédille	

3 — Le code de la lettre minuscule : 97 pour a

97	a
99	c
1Ø1	e
1Ø5	i
111	o
117	u

Ainsi "à" est représenté par les trois octets suivants : 22 65 97
Vérifiez-le :

```
?CHR$(22) + CHR$(65) + CHR$(97)
```

Pour écrire une minuscule accentuée au clavier, il vous faut d'ailleurs entrer successivement les trois "caractères" suivants : [ACC], l'accent, la minuscule.
Par exemple pour la lettre "à" :

[ACC]

[•] [Ø̀]

[A] en mode minuscule (c'est-à-dire avec la lampe allumée)

Pour entrer au clavier le caractère correspondant à l'accent, il faut appuyer simultanément sur :

[•] [CNT] [ω̂] pour le tréma

[•] [ω̂] pour l'accent circonflexe

[•] [7́] pour l'accent aigu

[•] [Ø̀] pour l'accent grave

Pour entrer au clavier le ç (c cédille), il faut appuyer sur [ACC] puis deux fois sur [C] en mode minuscule.

Autre exemple : pour taper "ï", il faut entrer successivement :

[ACC]

[•] [CNT] [ω̂]

[I] (en mode minuscule)

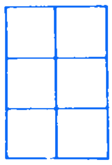
Voilà, c'est peut-être un peu long quelquefois (il faut reconnaître

que les "ï" ne sont pas si fréquents dans la langue française), mais ceci vous permettra d'écrire très correctement vos commentaires et vos textes sur l'écran, sans même une faute d'accent.

Les caractères semi-graphiques

Ce n'est pas tout.
Nous venons de voir le jeu de caractères "d'imprimerie", c'est-à-dire les caractères qui permettent d'écrire correctement un texte comme on le compose en imprimerie.
Mais notre ordinateur vous offre un peu mieux que cela : il dispose d'un ensemble de caractères destinés à faciliter la composition de dessins ou de schémas. On qualifie ces caractères de "semi-graphiques" par opposition avec le mode "graphique" qui dessine avec des points.

A l'écran, chaque caractère est inscrit dans un carré qui correspond en graphique à 8 points horizontaux et 8 points verticaux. On décompose ce caractère en 6 rectangles élémentaires que l'on peut "allumer" ou "éteindre" à volonté :



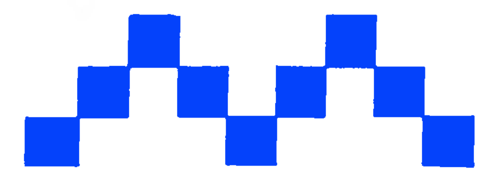
Chacun de ces caractères peut être affiché par son code en indiquant au préalable par un caractère particulier (CHR$(14)) que l'on veut passer en mode semi-graphique. Un autre caractère permet de revenir, si on le désire, du mode semi-graphique au mode graphique : CHR$(15).

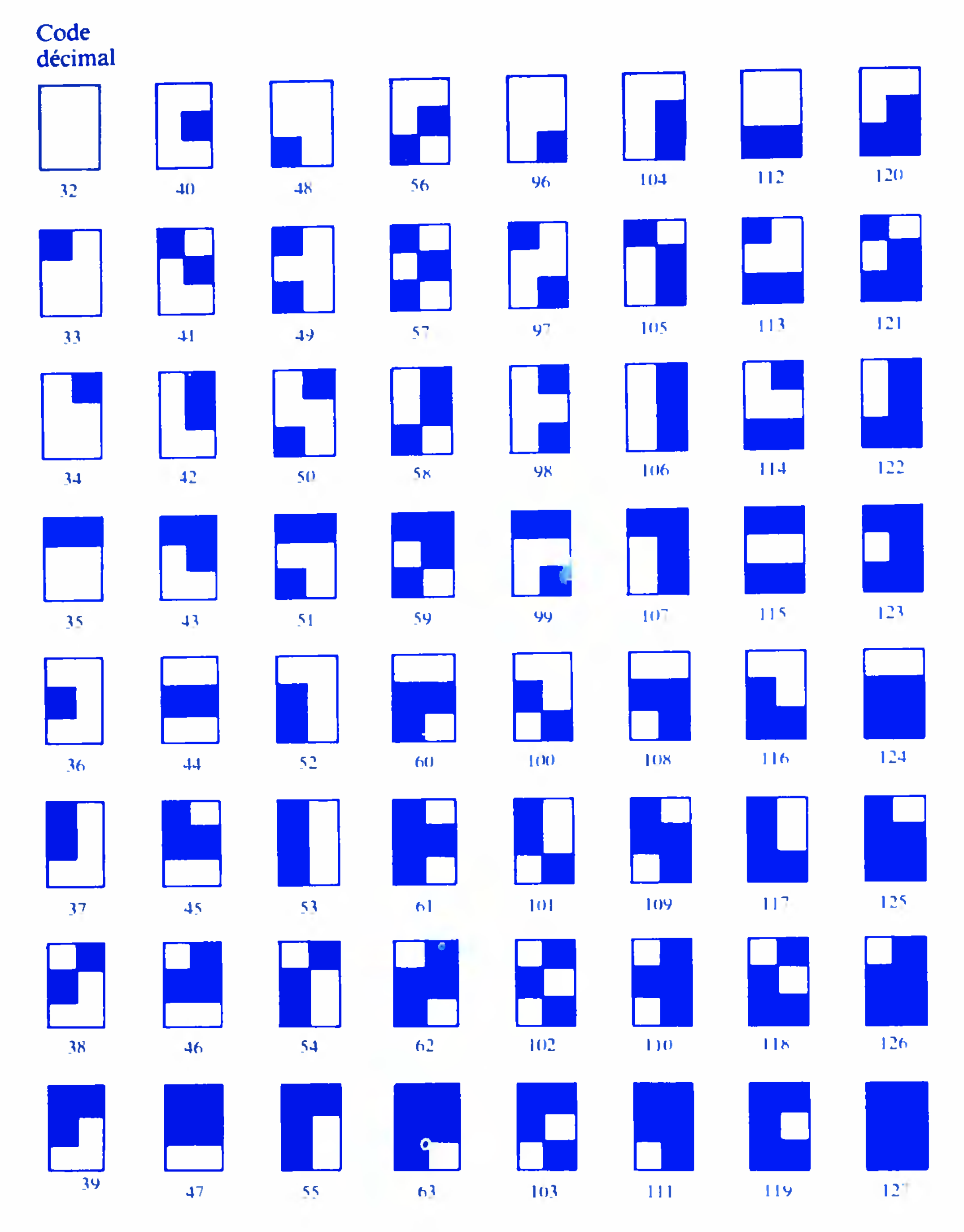
Essayez par exemple :

```
?CHR$(14)+CHR$(57)+CHR$(15)
```

Tous les caractères compris entre CHR$(14), qui fait passer en mode semi-graphique, et CHR$(15), qui fait revenir en mode graphique, sont considérés comme semi-graphiques.

```
?CHR$(14)+CHR$(56)+CHR$(41)+CHR$(56)+
CHR$(41)+CHR$(48)+CHR$(15)
```



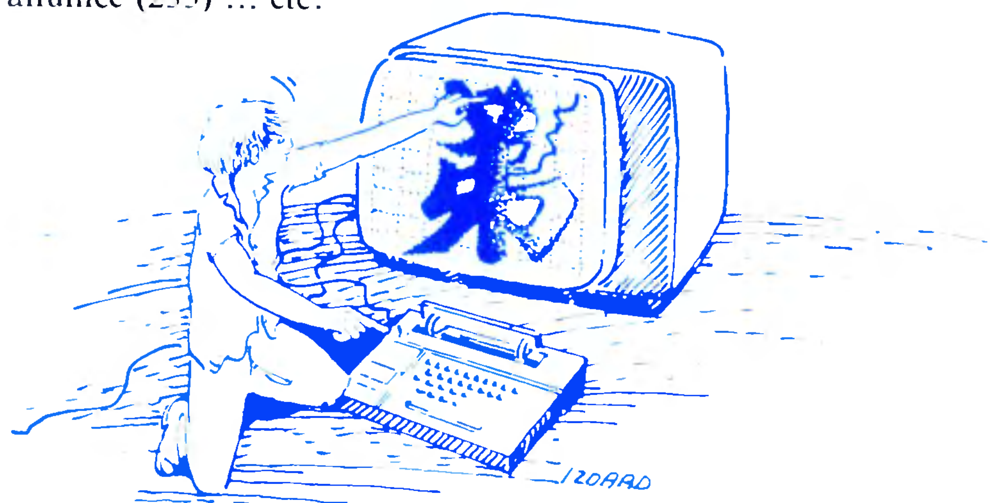


Des caractères à façon

En plus de tous les caractères que nous venons de voir, il vous reste encore la possibilité de dessiner vous-même vos caractères. Nous avons vu qu'un caractère est inscrit dans un carré de 8 lignes de 8 points.

Pour définir un caractère, il faudra indiquer pour chaque ligne

quels points sont "allumés" et quels points sont "éteints". La définition est faite par une instruction DEFGR$.
Par exemple :
DEFGR$(1) = 255, Ø, 255, Ø, 255, Ø, 255, Ø
définit le caractère numéroté 1 (on peut en créer jusqu'à 128) dont la première ligne en haut est complètement allumée (255), la deuxième complètement éteinte (Ø), la troisième complètement allumée (255) ... etc.



Pour expliquer comment on obtient le nombre correspondant à chaque ligne de points, il faudrait se livrer à une petite conversion de binaire en décimal. Afin de vous éviter ce travail, l'exemple n° 12 contient un programme qui fait la conversion automatiquement et vous donne la ligne d'instruction à écrire pour un caractère que vous dessinez avec le crayon optique.

Chaque caractère ainsi défini peut ensuite être employé en le désignant par GR$(n) où n est son numéro.
Il faut toutefois avant de commencer les définitions réserver la place de ces caractères en mémoire par une instruction CLEAR :
CLEAR,,3
réserve de la place en mémoire pour définir 3 caractères dont les numéros seront Ø,1 et 2 (ne pas oublier les deux virgules).
DEFGR$(1) = 255, Ø, 255, Ø, 255, Ø, 255, Ø
définit le caractère n° 1
?GR$(1)
affiche le caractère n° 1 formé de traits horizontaux
DEFGR$(Ø) = 85, 17Ø, 85, 17Ø, 85, 17Ø, 85, 17Ø
définit le caractère n° Ø

?GRS(Ø)

affiche le caractère n° Ø qui a la forme d'un damier très fin.

Vous pouvez maintenant exercer vos talents de dessinateur et créer ainsi les formes que vous voulez à l'écran. Il y a de nombreuses applications de ces caractères. En voici une qui pourra agrémenter les jeux de cartes que vous voudrez programmer : définir les caractères "pique", "cœur", "carreau", "trèfle". A vos programmes !

Retour sur la conversion binaire - décimal

Dans la définition d'un caractère, chaque ligne de points allumés ou éteints est représentée par un nombre compris entre Ø et 225. La codification se fait de la manière suivante :

— A chaque point, on associe la valeur binaire 1 s'il est allumé et Ø s'il est éteint.

— La ligne des 8 points est représentée par la valeur binaire des 8 points, c'est-à-dire par 8 chiffres Ø ou 1.

— On convertit alors cette valeur binaire en valeur décimale pour l'introduire ensuite dans l'instruction DEFGR$.

Prenons l'exemple d'une ligne où les points sont successivement allumés et éteints, en partant de la gauche. Sa valeur binaire sera :

1Ø1Ø1Ø1Ø

ce qui donne 17Ø en décimal.

Une ligne toute éteinte aura pour valeur binaire :

ØØØØØØØØ

soit Ø aussi en décimal.

Une ligne toute allumée aura pour valeur binaire :

11111111 soit 255 en décimal.

Voici pour vous aider une table de conversion de binaire en décimal :



Table de conversion binaire/décimal

Binaire	Décimal	Binaire	Décimal
ØØØØØØØØ	Ø	ØØ1ØØØØ1	33
ØØØØØØØ1	1	ØØ1ØØØ1Ø	34
ØØØØØØ1Ø	2	ØØ1ØØØ11	35
ØØØØØØ11	3	ØØ1ØØ1ØØ	36
ØØØØØ1ØØ	4	ØØ1ØØ1Ø1	37
ØØØØØ1Ø1	5	ØØ1ØØ11Ø	38
ØØØØØ11Ø	6	ØØ1ØØ111	39
ØØØØØ111	7	ØØ1Ø1ØØØ	40
ØØØØ1ØØØ	8	ØØ1Ø1ØØ1	41
ØØØØ1ØØ1	9	ØØ1Ø1Ø1Ø	42
ØØØØ1Ø1Ø	10	ØØ1Ø1Ø11	43
ØØØØ1Ø11	11	ØØ1Ø11ØØ	44
ØØØØ11ØØ	12	ØØ1Ø11Ø1	45
ØØØØ11Ø1	13	ØØ1Ø111Ø	46
ØØØØ111Ø	14	ØØ1Ø1111	47
ØØØØ1111	15	ØØ11ØØØØ	48
ØØØ1ØØØØ	16	ØØ11ØØØ1	49
ØØØ1ØØØ1	17	ØØ11ØØ1Ø	50
ØØØ1ØØ1Ø	18	ØØ11ØØ11	51
ØØØ1ØØ11	19	ØØ11Ø1ØØ	52
ØØØ1Ø1ØØ	20	ØØ11Ø1Ø1	53
ØØØ1Ø1Ø1	21	ØØ11Ø11Ø	54
ØØØ1Ø11Ø	22	ØØ11Ø111	55
ØØØ1Ø111	23	ØØ111ØØØ	56
ØØØ11ØØØ	24	ØØ111ØØ1	57
ØØØ11ØØ1	25	ØØ111Ø1Ø	58
ØØØ11Ø1Ø	26	ØØ111Ø11	59
ØØØ11Ø11	27	ØØ1111ØØ	60
ØØØ111ØØ	28	ØØ1111Ø1	61
ØØØ111Ø1	29	ØØ11111Ø	62
ØØØ1111Ø	30	ØØ111111	63
ØØØ11111	31	Ø1ØØØØØØ	64
ØØ1ØØØØØ	32	Ø1ØØØØØ1	65

Binaire	Décimal
Ø1ØØØØ1Ø	66
Ø1ØØØØ11	67
Ø1ØØØ1ØØ	68
Ø1ØØØ1Ø1	69
Ø1ØØØ11Ø	70
Ø1ØØØ111	71
Ø1ØØ1ØØØ	72
Ø1ØØ1ØØ1	73
Ø1ØØ1Ø1Ø	74
Ø1ØØ1Ø11	75
Ø1ØØ11ØØ	76
Ø1ØØ11Ø1	77
Ø1ØØ111Ø	78
Ø1ØØ1111	79
Ø1Ø1ØØØØ	80
Ø1Ø1ØØØ1	81
Ø1Ø1ØØ1Ø	82
Ø1Ø1ØØ11	83
Ø1Ø1Ø1ØØ	84
Ø1Ø1Ø1Ø1	85
Ø1Ø1Ø11Ø	86
Ø1Ø1Ø111	87
Ø1Ø11ØØØ	88
Ø1Ø11ØØ1	89
Ø1Ø11Ø1Ø	90
Ø1Ø11Ø11	91
Ø1Ø111ØØ	92
Ø1Ø111Ø1	93
Ø1Ø1111Ø	94
Ø1Ø11111	95
Ø11ØØØØØ	96
Ø11ØØØØ1	97

Binaire	Décima
Ø11ØØØ1Ø	98
Ø11ØØØ11	99
Ø11ØØ1ØØ	100
Ø11ØØ1Ø1	101
Ø11ØØ11Ø	102
Ø11ØØ111	103
Ø11Ø1ØØØ	104
Ø11Ø1ØØ1	105
Ø11Ø1Ø1Ø	106
Ø11Ø1Ø11	107
Ø11Ø11ØØ	108
Ø11Ø11Ø1	109
Ø11Ø111Ø	110
Ø11Ø1111	111
Ø111ØØØØ	112
Ø111ØØØ1	113
Ø111ØØ1Ø	114
Ø111ØØ11	115
Ø111Ø1ØØ	116
Ø111Ø1Ø1	117
Ø111Ø11Ø	118
Ø111Ø111	119
Ø1111ØØØ	120
Ø1111ØØ1	121
Ø1111Ø1Ø	122
Ø1111Ø11	123
Ø11111ØØ	124
Ø11111Ø1	125
Ø111111Ø	126
Ø1111111	127
1ØØØØØØØ	128

... etc.

Annexe 3/Un petit coup d'œil sur la mémoire

La mémoire libre

Nous avons vu au cours du chapitre XVII que la mémoire n'était pas extensible à souhait et qu'il n'est pas possible de déclarer, même si on ne les utilise pas entièrement, des tableaux de trop grande taille.
Rappelons que le moyen d'avoir une idée de la taille de l'espace disponible est fourni par l'instruction FRE(Ø) :

```
?FRE(Ø)
1515
```

La réponse indique 1515 octets libres dans l'espace qui vous est réservé, vous, c'est-à-dire l'utilisateur de BASIC. Il va de soi qu'au cours du développement d'un programme, vous obtiendrez rarement deux réponses identiques puisque cela dépend de ce que vous avez fait entre les deux demandes.
Par contre, il est certain que vous obtiendrez toujours le même nombre au départ, avant de rentrer un programme. Ce nombre dépend du volume de mémoire que vous avez acquis avec votre TO7 : 8 K, 24 K. Nous y reviendrons.



Agrandir l'espace réservé aux chaîne de caractères

La mémoire qui vous est attribuée, celle où se trouvent votre programme et les données qu'il utilise, comporte une partie spéciale

réservée au stockage des chaînes de caractères.
Essayez les quelques lignes suivantes :
A$ = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
B$ = A$ + A$ + A$
C$ = B$ + B$ + B$
?OS Error
L'espace fixé a priori pour les chaînes de caractères est de 3ØØ octets. En créant des chaînes de caractères trop longues, vous provoquez un message d'erreur OS (Out of String).

Pour connaître le volume de l'espace restant disponible pour les chaînes de caractères, il existe une variante de la fonction FRE : il suffit de mettre un nom, quelconque, de variable chaîne de caractère entre les deux parenthèses. Prenons par exemple X$:
?FRE(X$)
123
Si vous essayez :
?FRE(Y$)
123
Vous obtenez le même résultat.
Il est tout à fait possible que vous ayez besoin de plus de place pour mettre vos textes et moins pour les programmes (l'espace total est toujours le même).
Vous pouvez modifier le nombre fixé a priori (3ØØ) par une instruction CLEAR :
CLEAR 2ØØØ
réserve 2ØØØ octets pour les chaînes de caractères, ce qui soit dit en passant, laisse place à 2ØØØ caractères environ.

L'instruction CLEAR sert également à faire d'autres réservations : pour l'espace réservé aux programmes en assembleur (nous n'en sommes pas encore là) et pour le nombre de caractères semi-graphiques définis par l'utilisateur (ça, nous l'avons vu dans l'un des derniers exemples).

Notez encore que CLEAR remet à zéro toutes les variables numériques, vide toutes les variables chaînes et supprime tous les tableaux comme au moment de l'exécution par RUN.
Voilà pour CLEAR , c'est à peu près tout. Cette instruction aux multiples facettes est un peu délicate à manier mais elle constitue un moyen pratique d'adapter la mémoire (toujours trop petite) à ses besoins particuliers.

Organisation de la mémoire

Nous allons examiner ici la géographie de la mémoire. Cette démarche d'explorateur va nous obliger à employer quelques nombres et quelques termes ésotériques (nous nous excusons auprès de notre aimable clientèle qui pourra trouver dans d'autres ouvrages des compléments et des développements très intéressants sur les microprocesseurs et autres micro-ordinateurs).
La suite est réservée à ceux qui veulent "bricoler" avec leur ordinateur pour voir ce qu'il a dans le ventre et n'est pas du tout indispensable pour rédiger ses programmes.

L'espace total utilisable par le microprocesseur situé dans le TO7 s'élève à 64 Koctets soit exactement 65536 octets. Pour plus de commodités, l'emplacement (l'"adresse") de chacun de ces octets est repéré par un nombre écrit dans une base hexadécimale où l'on utilise 16 "chiffres" différents notés : Ø, 1, 2, 3, 4, 5, 6, 7, 8 , 9, A, B, C, D, E, F.
Le premier octet a pour adresse ØØØØ, le dernier FFFF.

En commençant par le début, on trouve :

Adresses	Zone
ØØØØ 3FFF	zone cartouche BASIC
4ØØØ 5FFF	mémoire écran
6ØØØ 6ØFF	réservé au système
61ØØ DFFF	libre pour mémoire utilisateur
EØØØ FFFF	système

L'interpréteur BASIC, en mémoire morte (c'est-à-dire inaccessible pour l'utilisateur), occupe les 16 premiers K octets de la mémoire.
La mémoire qui contient le dessin de l'écran réside dans les 8 Koctets suivants.

Les quelques variables utilisées par le système se situent ensuite, en 6ØØØ.
Puis à partir de 61ØØ, commence la zone utilisateur, celle qui contient vos programmes et vos données numériques et chaînes de caractères.
Le moniteur, c'est-à-dire le programme qui contrôle l'écran, le clavier, le magnétophone, ... se situe vers la fin de la mémoire, avec au milieu, l'emplacement de quelques octets pour les circuits d'interfaces avec les périphériques.

Comment jeter un coup d'œil sur un octet, ... et y écrire



Il ne servirait pas à grand'chose de savoir comment est organisée la mémoire s'il n'était pas possible d'aller y regarder de plus près.
Pour ce faire, il existe une fonction spéciale réservée aux plus curieux : PEEK

```
?PEEK (26392)
37
```

La réponse 37 est le nombre (décimal) contenu dans l'octet numéro 26392.
Ce numéro 26392 (nombre décimal) est l'adresse de l'octet.
Le contenu de cet octet peut évidemment varier suivant le programme et les données présents au moment où vous avez tapé PEEK.
En manipulant l'instruction PEEK avec habileté et persévérance vous pourrez explorer à votre aise la mémoire de l'ordinateur et vous rendre compte de la manière dont sont conservés programmes et données.

Ayant vu ce que contient un octet, comment modifier ce contenu ?
C'est le rôle de l'instruction POKE :
POKE 26392, 38
a pour effet d'inscrire le nombre (décimal) 38 dans l'octet 26392.
Cette instruction a donc une action de modification de la mémoire. Si l'ancienne valeur 37 avait une signification bien précise, il faut que la nouvelle en ait une qui soit tout aussi cohérente avec le reste. Sinon, votre programme risque fort d'être un peu perturbé ou d'aller se planter ... dans les choux.

Ne craignez rien, le pire qui puisse vous arriver est de détruire le programme résident en mémoire à ce moment-là ; vous ne risquez pas de détruire votre BASIC avec des POKE. Et avec un peu d'habitude l'usage de POKE vous permettra peut-être des modifications astucieuses.

Annexe 4/Fonctions mathématiques

Avec les fonctions mathématiques suivantes, vous pourrez faire les calculs les plus courants :

ABS(X) donne la valeur absolue de X :

```
PRINT ABS(89)
89
PRINT ABS(-21)
21
```

SGN(X) donne le signe de X multiplié par 1 :

si X = Ø alors SGN(X) = Ø

si X>Ø alors SGN(X) = 1

si X<Ø alors SGN(X) = -1

INT(X) donne le premier entier immédiatement inférieur à X :

```
PRINT INT(3.21)
3
PRINT INT(-3.21)
-4
```

FIX(X) supprime les chiffres après la virgule :

```
PRINT FIX(3.21)
3
PRINT FIX(-3.21)
-3
```

SQR(X) donne la racine carrée de X (si X est négatif, on obtient un message d'erreur)

```
PRINT SQR(142129)
377
```

LOG(X) donne le logarithme népérien (à base *e*) de X

```
PRINT LOG(1Ø)
2.3Ø259                    (valeur arrondie à 6 chiffres)
PRINT LOG(2.7182818)
1
```

EXP(X) donne la valeur exponentielle de X c'est-à-dire *e* à la puissance X

```
PRINT EXP(1)
2.71828                    (valeur arrondie à 6 chiffres)
```

SIN(X) donne le sinus de X (X étant exprimé en radians)

COS(X) donne la cosinus de X

TAN(X) donne la tangente de X

RND donne un nombre au hasard compris entre Ø et 1

Annexe 5/Mots réservés du BASIC

Le nom d'une variable ne peut commencer par l'un des mots suivants :

ABS
AND
ASC
ATTRB
AUTO

BEEP
BOX

CDBL
CHR$
CINT
CLEAR
CLOSE
CLS
COLOR
CONSOLE
CONT
COS
CSNG
CSRLIN

DATA
DEF
DELETE
DIM

ELSE
END
EOF
EQV
ERL
ERR
ERROR
EXEC
EXP

FIX
FN
FOR
FRE

GO
GR$

HEX$

IF
IMP
INKEY$
INPEN
INPUT
INSTR
INT

LEFT$
LEN
LET
LINE
LIST
LOAD
LOCATE
LOG

MERGE
MID$
MOD
MOTOR

NEW
NEXT
NOT

OCT$
OFF
ON
OPEN
OR

PEEK
PEN
PLAY
POINT
POKE
POS
PRINT
PSET
PTRIG

READ
REM
RESTORE
RESUME
RETURN
RIGHT$
RND
RUN

SAVE
SCREEN
SGN
SIN
SKIPF
SPC(
SQR
STEP
STICK
STOP
STRIG
STR$
SUB

TAB
TAN(
THEN
TO
TROFF
TRON

UNMASK
USING
USR

VAL
VARPTR

WAIT
WHILE
WEND

XOR

Annexe 6/Liste des principaux codes d'erreur

Voici la liste alphabétique des principaux codes d'erreur que vous pouvez rencontrer. Ces codes sont fabriqués en prélevant deux lettres dans un court message qui explicite le type de l'erreur. Vous trouverez ici le code à deux lettres, le message qui y correspond et une explication de l'erreur :

BS **Bad Subscript**
L'indice d'un élément de tableau dépasse la valeur maximum ou bien est négatif.

CN **Can't Continue**
La commande CONT ne permet pas de poursuivre l'exécution du programme.

DD **Duplicate Definition**
Il n'est pas possible de déclarer deux fois le même tableau avec l'instrution DIM .

FC **Illegal Function Call**
Une fonction a été appelée avec des valeurs interdites. Exemple : PSET (– 1, – 1).

FN **For without Next**
Aucun NEXT n'a été rencontré après l'exécution d'un FOR .

ID **Illegal Direct**
L'instruction ne peut pas être utilisée en mode direct, c'est-à-dire en dehors d'un programme. C'est le cas de INPUT .

IO **Device Input Output Error**
Un périphérique ne fonctionne pas bien, ou est mal connecté : cassette, imprimante, ...

LS **String too Long**
Une chaîne de caractères ne peut pas dépasser 255 caractères.

MO **Missing Operand**
Il manque un opérande à une opération comme une addition, soustraction, ...

NF **Next without For**
Une instruction NEXT a été rencontrée alors que le FOR correspondant n'a pas été exécuté.

OD **Out of Data**
Il n'y a pas assez de données en DATA pour le nombre de READ à exécuter.

OM **Out of Memory**
Il n'y a pas assez de place en mémoire pour tout ce qu'on veut y faire : déclarer un tableau, imbriquer plusieurs boucles, ...

OS **Out of String**
Il n'y a pas assez de place pour toutes les chaînes de caractères. On peut modifier la taille de cette zone par une instruction CLEAR .

OV **Overflow**
L'ordinateur est débordé : le nombre qu'il doit lire ou calculer est trop grand.

RG **Return without Gosub**
L'instruction RETURN a été rencontrée alors que l'appel de sous-programme n'a pas été fait par GOSUB .

SN **Syntax Error**
Erreur de syntaxe très fréquente. Elle survient à chaque fois qu'une instruction est mal orthographiée, qu'il manque une virgule, un espace, une parenthèse, etc.

TM **Type Mismatch**
Il n'est pas possible de mettre un nombre dans une variable chaîne de caractère ou inversement.

UL **Undefined Line Number**
On ne peut pas faire de GOTO ou de GOSUB à une ligne qui n'existe pas.

Ø **Division par zéro**

Annexe 7/Résumé des commandes, instructions et fonctions

ASC("F")

donne le numéro de code ASCII correspondant au caractère "F", soit 7Ø.

ATTRB Ø, 1

donne à l'écran des caractères en double hauteur.

ATTRB 1, Ø

donne à l'écran des caractères en double largeur.

ATTRB 1, 1

donne à l'écran des caractères en double hauteur et double largeur.

BEEP

exécute un "bip" dans le haut-parleur du téléviseur.

BOX (1Ø, 1Ø) - (1ØØ, 2ØØ)

dessine un rectangle de coins opposés (1Ø, 1Ø) et (1ØØ, 2ØØ).

BOX (1Ø, 1Ø) - (1ØØ, 2ØØ), 2

dessine le même rectangle en vert (couleur 2).

BOXF (1Ø, 1Ø) - (1ØØ, 2ØØ), 2

dessine le même rectangle "en plein".

CHR$ (87)

c'est le caractère dont le code ASCII est 87, soit W.

CLEAR 2ØØØ

réserve 2ØØØ octets (au lieu du minimum de 3ØØ) pour les chaînes de caractères.

CLS

efface entièrement l'écran.

[CNT] C

interrompt l'exécution d'un programme.

COLOR 1, 2

affecte la couleur 1 aux caractères (et aux dessins), affecte la couleur 2 au fond des caractères.

CONSOLE 2

réserve les 2 premières lignes de l'écran : l'affichage se fera sur les 23 autres lignes.

CONT

permet de reprendre l'exécution d'un programme après arrêt par [CNT] C.

DATA 4, "DISQUES", 3ØØ

est une liste de données numériques ou chaînes de caractères (les guillemets sont facultatifs pour les chaînes de caractères).

DELETE 1ØØ - 2ØØ

efface le programme de la ligne 1ØØ à la ligne 2ØØ.

DELETE 15Ø

efface seulement la ligne 15Ø.

DELETE 15Ø

efface de la ligne 15Ø jusqu'à la fin.

DIM A (2Ø)

réserve de la place en mémoire pour un tableau de 21 nombres : A (Ø), A (1), ... A (2Ø).

DIM A$ (2Ø)

réserve de la place pour un tableau de 21 chaînes de caractères : A$ (Ø), A$ (1), ... A$ (2Ø).

END

indique la fin du programme.

FOR T = 1 TØ 1Ø
NEXT T

les instructions situées entre FOR ... et NEXT T doivent être exécutées 1Ø fois de suite.
Le pas de variat on de la variable T peut être différent de 1 :
FOR T = 1 TO 1Ø STEP 2
dans ce cas T prend les valeurs 1, 3, 5, 7, 9 et la « boucle » est exécutée 5 fois (après la dernière boucle T vaut 11).
FOR W = 1ØØ TO 5Ø STEP –1Ø
dans ce cas W prend les valeurs 1ØØ, 9Ø, 8Ø, 7Ø, 6Ø, 5Ø et la boucle est exécutée 6 fois.

FRE (Ø)

donne le nombre d'octets libres en mémoire.

GOSUB 1ØØØ

oblige le programme à aller à la ligne 1ØØØ, première ligne d'un sous-programme. La dernière ins-

truction du sous-programme est RETURN : elle fait reprendre l'exécution à la ligne qui suit l'instruction GOSUB .

GOTO 2ØØ

branche le programme à la ligne 2ØØ.

GR$ (3)

est le caractère numéro 3 défini par l'utilisateur PRINT GR$ (3) affichera ce caractère à l'écran.

IF A > 3 THEN 1ØØ

IF NOM$ = "XXX" THEN PRINT "TERMINE"

si la condition qui suit IF est réalisée, alors l'instruction qui suit THEN est exécutée (branchement à une certaine ligne ou autre instruction). Sinon le programme saute ce qui suit THEN et poursuit à la ligne suivante.

INKEY$

sert à introduire dans l'exécution d'un programme un caractère frappé au clavier (sans avoir à utiliser INPUT).
A$ = INKEY$ affecte à la variable A$ la lettre frappée au clavier.
Si aucune lettre n'est frappée, le programme continue avec A$ vide.

INPUT R

affiche un point d'interrogation à l'écran et attend une réponse numérique qui sera affectée à la variable R.

INPUT NOM$

même chose avec une variable chaîne de caractères.

INPUT "JOUR, MOIS, ANNEE" ; J, M$, A

affiche la phrase entre guillemets et attend 3 réponses (un nombre, une chaîne de caractères, un nombre). Au clavier les réponses successives doivent être séparées par des virgules.

INPUTPEN C, L

affecte aux variables C et L les numéros de colonne et de ligne du point désigné à l'écran avec le crayon optique.

INT (A)

donne le nombre entier immédiatement inférieur à A :
INT (13.63) = 13
INT (Ø.74) = Ø.

LEFT$ (A$, I)

donne les I premiers caractères de la chaîne A$ (à partir de la gauche).

LINE (5Ø, 1ØØ) - (2ØØ, 15Ø), 4

trace le trait qui joint le point (5Ø, 1ØØ) au point (2ØØ, 15Ø) dans la couleur 4 (bleu).

LINE - (1ØØ, 1ØØ)

joint le dernier point tracé à l'écran au point (1ØØ, 1ØØ).

LIST

affiche à l'écran la suite des lignes d'instruction constituant le programme.

LIST 1ØØ-2ØØ

affiche les lignes 1ØØ à 2ØØ.

LIST -3ØØ

jusqu'à la ligne 3ØØ.

LIST 2ØØ-

depuis la ligne 2ØØ jusqu'à la fin.
Il est possible d'arrêter le déroulement d'un listing à l'écran en appuyant sur la touche STOP et le reprendre en appuyant sur n'importe quelle touche; on peut aussi l'arrêter définitivement avec CNT C.

LIST "LPRT :" liste le programme sur l'imprimante.

LOAD "TELECRAN"

charge dans l'ordinateur le programme TELECRAN conservé sur cassette. Avant d'envoyer cette instruction, il faut positionner la bande et appuyer sur le bouton "lecture" du magnétophone. Si le programme cherché n'est pas trouvé, la lecture ne peut être arrêtée que par appui sur la touche de réinitialisation.

LOCATE 1Ø, 5

place le curseur en colonne 1Ø et ligne 5.

LOCATE 1Ø, 5, Ø

même chose avec le curseur invisible.

MID$ ("TINTIN ET MILOU", 11)

est la sous-chaîne extraite à partir du 11e caractère, soit MILOU .

MID$ ("TINTIN ET MILOU", 11, 3)

est la sous-chaîne de longueur 3 extraite à partir du 11e caractère, soit : MIL.

NEW

efface le programme en mémoire (et vide toutes les variables).

NEXT

voir FOR ... NEXT .

OR

c'est toujours de l'anglais ! et cela signifie "ou"
IF A < 1Ø OR A > 1ØØ THEN PRINT A

ON TEST GOTO 12Ø, 2ØØ, 3ØØ

cette instruction ON ... GOTO ... permet de brancher le programme à
la ligne 12Ø si la variable TEST vaut 1
la ligne 2ØØ si la variable TEST vaut 2
la ligne 3ØØ si la variable TEST vaut 3.
(On peut bien sûr utiliser plus de 3 branchements.)

ON W GOSUB 1ØØØ, 15ØØ, 3ØØØ

cette instruction fonctionne comme ON ... GOTO ... : elle provoque le branchement au 1er, 2e, 3e... sous-programme suivant les valeurs de la variable W.
Mais après avoir traité le sous-programme, l'exécution reprend immédiatement après la ligne ON ... GOSUB .

ONPEN GOTO 1ØØ, 2ØØ, 6ØØ

ONPEN GOSUB 1ØØØ, 3ØØØ

branche le programme sur les numéros de lignes en fonction du numéro de la zone désignée à l'écran avec le crayon lumineux (à la zone n° Ø correspond la première adresse, ...).

PEN Ø ; (Ø, Ø) - (5Ø, 5Ø), 1 ; (Ø, 5Ø) - (5Ø, 1ØØ)

définit 2 zones rectangulaires à l'écran : la première (Ø, Ø) - (5Ø, 5Ø) et la deuxième (Ø,5Ø) - (5Ø, 1ØØ).

On peut ainsi définir jusqu'à 8 zones différentes (éventuellement avec plusieurs instructions PEN successives).

Cette instruction est utilisée couplée avec ONPEN ... GOTO ou ONPEN ... GOSUB .

PLAY "DOREMI"

joue la suite des notes DO, RE, MI dans le haut-parleur.

PRINT M$

affiche la chaîne de caractères placée dans la variable M$: si M$ est "JANVIER", l'ordinateur affiche JANVIER.

PRINT "DATE :"

affiche le texte entre guillemets soit DATE: .

PRINT "DATE:" ; A ; M$

le point-virgule permet d'afficher plusieurs nombres ou chaînes de caractères les uns à la suite des autres. Ici : DATE : 12 JANVIER.

PSET (15Ø, 5Ø), 1

affiche un point à l'écran en colonne 15Ø et ligne 5Ø, dans la couleur 1 (rouge).

READ A, D, X$

affecte aux variables A, D, X$ les valeurs correspondantes données dans l'instruction DATA (la première donnée est affectée à A, la deuxième à D, ...)

REM COMMENTAIRE DU PROGRAMME
' COMMENTAIRE DU PROGRAMME

tout ce qui est après l'instruction REM (ou une apostrophe) est ignoré par l'ordinateur.

RETURN

voir GOSUB .

RIGHT$ ("VOLTAIRE", 5)

donne la sous-chaîne formée par les 5 caractères les plus à droite de la chaîne donnée : TAIRE.

RND

donne un nombre au hasard compris entre Ø et 1.

RUN

déclenche l'exécution du programme (et vide toutes les variables).

RUN 2ØØ

déclenche l'exécution à partir de la ligne 2ØØ.

SAVE "TELECRAN"

enregistre le programme "TELECRAN" qui est en mémoire sur cassette. (Le magnétophone doit être en mode enregistrement).

SCREEN 7, 6, 4

fixe les couleurs des caractères : 7 (noir), du fond : 6 (bleu-clair), du pourtour de l'écran : 4 (bleu foncé).

TAB

permet une présentation en tableau à l'écran : PRINT TAB (1Ø) positionne le curseur sur la colonne 1Ø.

TRON

affiche les numéros de lignes au fur et à mesure qu'elles sont exécutées lors du déroulement du programme.

TROFF

supprime l'effet de l'instruction TRON .

VAL (X$)

donne la valeur numérique d'une chaîne de caractères :
— si X$ = "356" alors VAL (X$) = 356,
— si X$ = "356 FRANCS" alors VAL (X$) = 356,
— si X$ commence par un caractère qui n'est ni un nombre, ni un blanc, ni un signe suivi d'un chiffre, alors VAL (X$) = Ø.

Annexe 8 / Utilisation d'une imprimante et d'autres périphériques

L'imprimante

Si votre TO7 ou TO7-70 est équipé d'une imprimante et de l'interface de communication, vous allez pouvoir imprimer la liste de vos programmes et bien d'autres choses encore.
Tout d'abord, n'oubliez pas de brancher l'imprimante et de l'allumer avant l'unité centrale ; l'inverse n'est jamais bon pour la mémoire et son contenu.
Pour lister un programme, l'ordre à donner est semblable :
LIST "LPRT :"
où LPRT : est le nom de code de l'imprimante pour le BASIC (Line PRinTer).
Cette commande fait sortir la liste du programme, si vous avez mis du papier.
Pour obtenir une liste partielle, par exemple de la ligne 3Ø à la ligne 2ØØ, il faut taper :
LIST "LPRT :", 3Ø-2ØØ
Suivant le type d'imprimante, la largeur du papier peut contenir 40 ou 80 caractères et même plus.
On indique le nombre de caractères par ligne en le mettant entre parenthèses :
LIST "LPRT : (80)"
pour une imprimante à 40 caractères.
Si vous avez une imprimante thermique (Référence PR 90.040), ou une imprimante matricielle (référence 90.582), vous pouvez obtenir une copie sur papier de vos belles images à l'écran. Il suffit de taper la commande :
SCREEN PRINT
vous pourrez ainsi garder un souvenir de vos plus belles créations par ordinateur.

Annexe 9 / Les spécialités du TO7-70

Les couleurs

Le TO7-70 possède 16 couleurs aussi bien pour le fond de l'écran que pour la forme (dessin et caractères).
Ces couleurs sont : les huit couleurs du TO7 auxquelles s'ajoutent huit nouvelles couleurs, proches des huit premières, mais plus claires.
Le BASIC, identique pour le TO7 et le TO7-70, ne connaît que huit couleurs, en particulier dans les instructions SCREEN et COLOR. Pour faire apparaître les huit nouvelles couleurs, il faut envoyer quelques caractères spéciaux.
En prenant deux variables F et C pour représenter les diverses options possibles :

si F = 112 pour les caractères
= 120 pour le fond
= 128 pour le tour de l'écran

Si C = Ø pour gris
1 pour rose
2 pour vert clair
3 pour jaune poussin
4 pour bleu ciel
5 pour rose parme
6 pour cyan clair
7 pour orange

On pourra obtenir le même résultat que SCREEN pour la couleur des caractères et du fond par :

PRINT CHR$ (22) CHR$ (32) CHR$ (F + C)

Ainsi, pour avoir des caractères (et des dessins) rose parme sur tout l'écran, il faut faire :

PRINT CHR$ (22) CHR$ (32) CHR$ (112 + 5)

Par contre, si l'on veut un fond orange, il faut écrire :

PRINT CHR$ (22) CHR$ (32) CHR$ (120 + 7)

On obtiendra le même résultat que SCREEN pour la couleur du tour de l'écran par :

PRINT CHR$ (22) CHR$ (F + C) avec F = 128

ce qui donne, pour un tour vert clair :

PRINT CHR$ (22) CHR$ (128 + 2)

Pour modifier uniquement la couleur des caractères ou du fond

de ce qui va être visualisé par la suite, comme dans l'instruction COLOR, il faut écrire
PRINT CHR$ (22) CHR$ (F+C)
Ainsi pour écrire la suite en rose parme :
PRINT CHR$ (22) CHR$ (112+5)
ou encore, pour écrire sur fond vert clair :
PRINT CHR$ (22) CHR$ (120+2)
Le fait de modifier la couleur des caractères de cette manière aura une incidence sur celle des dessins. La séquence d'instructions suivante :
PRINT CHR$ (22) CHR$ (112+5); "ABCDEF" : LINE (50, 100) (100,100)
a pour effet de visualiser ABCDEF en rose parme et la ligne horizontale 100 en rouge. Les tracés graphiques ne prennent que les huits couleurs de base, celles qui se rapprochent de ce qui a été fixé par le PRINT précédent.

La mémoire

L'organisation de la mémoire du TO7-70 est identique à celle du TO7. Mais vous disposez dès le départ de 32 K octets directement utilisables en BASIC.
Une zone supplémentaire de 16 K est également disponible, sans extension mémoire, mais aux mêmes emplacements que la moitié des 32 K précédents. Un petit programme en assembleur, pour les plus calés, permet d'accéder à cette zone. Nous ne l'aborderons pas ici ; tout est indiqué dans le manuel de référence BASIC de TOTEK.

L'impression de ce livre
a été réalisée sur les presses
des Imprimeries Aubin
à Poitiers Ligugé



Achevé d'imprimer le 30 novembre 1984
N° d'impression, P 13155
Dépôt légal, novembre 1984

Imprimé en France.

INDEX